Universidade de Aveiro Departamento de Engenharia Mecânica
2006

Hugo Miguel Gomes Alvarinhas

Localizar, Negociar e Contratar Serviços na WEB

Universidade de Aveiro Departamento de Engenharia Mecânica
2006

Hugo Miguel Gomes Alvarinhas

Localizar, Negociar e Contratar Serviços na WEB

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Mecânica – Área de Especialização em Automação e Robótica, realizada sob a orientação científica do Prof. Doutor José Paulo Oliveira Santos, Professor Auxiliar do Departamento de Engenharia Mecânica da Universidade de Aveiro

Dedicatória *A Meus Pais…*

O Júri

Presidente	Doutor José Joaquim de Almeida Grácio, Professor Catedrático da Universidade de Aveiro
Vogais	Doutor Carlos Baptista Cardeira, Professor Auxiliar do Instituto Superior Técnico da Universidade de Lisboa
	Doutor José Paulo Oliveira Santos, Professor Auxiliar da Universidade de Aveiro

Agradecimentos

Em primeiro lugar quero agradecer a meus pais pela perpetuação da vida, por todo o amor, carinho, preocupação, educação, companheirismo, pelos bons conselhos que me deram ao longo de toda a minha vida. Pela força, apoio e estímulo em começar e concluir este trabalho de dissertação. Por estarem a meu lado nos momentos que mais precisei. Muito obrigado D. Leonor e Sr. Alvarinhas, isto é, Mãe e Pai

Quero agradecer à minha irmã pelo apoio e companheirismo demonstrados desde sempre

Um agradecimento também muito especial ao meu orientador Professor Doutor José Paulo Oliveira Santos pela disponibilidade, incentivo, conhecimento transmitido e pelo aconselhamento e orientação desta dissertação de mestrado

Um agradecimento especial à Ana Baptista, uma pessoa também ela muito especial, pelo amor e carinho, pelo incentivo e estímulo em concluir esta dissertação

Um agradecimento profissional e particular à RENAULT CACIA, em especial ao Departamento de Manutenção, pela compreensão, colaboração, transversalidade e pela disponibilidade que me proporcionou para a realização deste trabalho

Um agradecimento também especial aos hotéis TURISMO e O ALAMBIQUE que permitiram que a música tivesse feito parte deste momento da minha vida

Sem querer especificar nem esquecer ninguém agradeço a TODOS os que, de uma maneira ou de outra, me deram o apoio e carinho especiais nesta fase da minha vida

Palavras-chave

Web Services, SOAP, UDDI, WSDL, Broker, Comércio Electrónico

Resumo

Desde muito cedo, com o crescente nível de competitividade das empresas urgiu a necessidade destas recorrerem às várias tecnologias existentes para, de uma forma mais eficaz e eficiente, divulgarem os seus produtos e serviços para poderem ser contratadas ou subcontratarem serviços a outras empresas. O uso de telefone, fax e carta foi-se tornando um meio obsoleto de localizar fornecedores, solicitar orçamentos, formalizar e realizar encomendas. A Internet, a cada dia que passa, vem-se afirmando como um meio privilegiado e indispensável para as empresas trocarem e disponibilizarem de uma forma rápida e eficaz todo o tipo de dados. O pagamento de serviços, de uma forma automática e em tempo real, permite também potenciar a concretização do chamado comércio electrónico, evitando dificuldades burocráticas no pagamento a fornecedores e entidades bancárias de e para qualquer parte do mundo. Neste sentido, este trabalho propõe um motor de busca capaz de fornecer às entidades contratantes uma forma simples, rápida e automática de adquirir produtos através da WEB com o mínimo de intervenção humana. Com base em requisitos predefinidos, este motor de busca é capaz de localizar automaticamente as entidades capazes de fornecer um produto ou serviço e solicitar-lhes orçamentos. De uma forma totalmente autónoma e sem intervenção humana, as entidades contactadas respondem com um orçamento e o motor de busca analisa todos os orçamentos recebidos, apresentando no final o orçamento temporalmente ou economicamente mais vantajoso. Disponibiliza ainda serviços que permitem formalizar e realizar encomendas e fazer o seu seguimento, comportando-se assim como interlocutor entre entidades contratantes e contratadas, desde o início da encomenda até à sua conclusão.

De uma forma genérica, este motor de busca chamado de Broker, tem como objectivo encontrar, em todo o mundo, uma entidade passível de ser subcontratada (Produtor) e que produza um determinado produto ao mais baixo preço ou o mais rapidamente possível. Com a WEB como meio de transmissão, são utilizadas as linguagens de programação PHP e HTML e o servidor WEB APACHE. São propostos serviços WEB baseados nas especificações SOAP, WSDL e UDDI. Neste trabalho é também proposta uma plataforma computacional a implementar em cada um dos produtores de forma a poderem disponibilizar um centro de maquinação numa rede computacional distribuída.

No final deste trabalho foi possível implementar um serviço autónomo e automático de negociação de serviços, realização e seguimento de encomendas. Para implementar a solução proposta foram também utilizados os protocolos LSV2 para aceder e controlar o centro de maquinação, as ligações ODBC e comandos SQL para aceder às bases de dados utilizadas no Broker e nos Produtores.

As empresas contratantes passam a dispor de uma aplicação, baseada na WEB, capaz de localizar e contactar um grande número de entidades a nível mundial, negociar e fornecer orçamentos, realizar e fazer o seguimento de encomendas de uma forma automática, instantânea e sem depender da disponibilidade nem da intervenção humana. Deixa de ser necessário elaborar documentos para solicitar orçamentos e formalizar encomendas e são reduzidos os tempos de espera e gastos no envio destes documentos. As entidades passíveis de serem contratadas são capazes de, automaticamente e sem intervenção humana, fornecer orçamentos e receber encomendas nas suas bases de produção, assim como disponibilizar os seus recursos fabris na WEB.

Keywords

Web Services, SOAP, UDDI, WSDL, Broker, e-business

Abstract

To increase the level of competitiveness of the companies, existing technologies are being appealed since very early, as mechanisms to efficiently divulge its products or services. This way, the companies are able to be contracted or to contract services to other companies. The technologies as telephone, fax and letter have been becoming an obsolete way to locate, to request budgets, to legalize and to carry through orders. Internet is being considered as a privileged and indispensable way for companies to give itself to know and to share its products. Allowing the real time and automatic payment, Internet increases the implementation of the already called e-business, preventing payment barriers to suppliers and banking entities from and to any place of the world.

This work, intends to present one intelligent, and human free search engine capable to supply a set of world localization WEB sites, allowing the acquisition of products requested by contracting entities. Based on a set of predefined requisites, this engine has an autonomous and automatic capacity to locate and to request budgets to all entities that are able to supply the intended products. It has also the capacity to analyze all the received companies' budgets, presenting to the contracting entity the fastest and economically more advantageous budget. It also permits to order and to follow the production state, behaving as interlocutor between the contracting entity and contracted entities, since the beginning's order until its conclusion.

Basically, this search engine called *Broker*, has as objective to find automatically, in the whole world, one entity that can supply one product at the smallest price and as quickly as possible. It uses the WEB platform, PHP and HTML pages and has APACHE as WEB Server. WEB services are proposed based on SOAP, WSDL and UDDI. This work also presents a computational platform that will be implemented in each one of the suppliers which will permit to share and manage a production of a milling machine in one distributed computational network. LSV2 Protocol is presented as mechanism that permits the control of milling machines in the proposed scenery. SQL and ODBC link are also used as a way to accede and to manage the Broker and Suppliers productions databases. At the end, this work has made possible to implement an autonomous and automatic mechanism to negotiate, formalize and follow order services.

The contracting companies start to make use of a WEB based application, capable to locate and contract a large number of world-wide level entities.

Contracting companies will be able and to, automatically, instantaneity and without depending neither on the availability or the intervention human, supply budgets, negotiate, formalize and make the pursuit of orders.

Bureaucratic problems related with the payment of services are prevented, starting to be an easier and fast process, elaboration of documents to request budgets and to legalize orders are no more needed and it is reduced the expenses and expedition times of these requesting documents. The entities to be contracted are capable to, automatically and without human intervention, supply budgets and to receive orders in its production databases, sharing theirs resources in the WEB.

Índice

1 Introdução 3
1.1 O Problema 4
1.2 Objectivos 5
1.3 Situação Actual 6
1.4 Solução Proposta 6
1.5 Organização da Dissertação 8

2 Estado da arte 13
2.1 Tecnologias de Suporte 13
2.1.1 DCE 15
2.1.1.1 Evolução Histórica 15
2.1.1.2 Arquitectura 15
2.1.2 CORBA 19
2.1.2.1 História e Evolução 19
2.1.2.2 Arquitectura 20
2.1.3 OPC 22
2.1.3.1 OPC – DA (Data Access) Custom Interface Standard (v3.00) 22
2.1.3.2 OPC-XML-DA 22
2.1.4 WEB Services 23
2.1.4.1 SOAP 26
2.1.4.1.1 História e Evolução 26
2.1.4.1.2 Estrutura das Mensagens SOAP 27
2.1.4.2 WSDL 28
2.1.4.2.1 História e Evolução. 28
2.1.4.2.2 Estrutura dos Documentos WSDL 29
2.1.4.3 UDDI 30
2.1.4.3.1 História e Evolução 31
2.1.4.3.2 Estrutura do UDDI 31
2.1.4.3.3 Métodos do UDDI 37
2.1.5 Considerações Sobre as Tecnologias de Suporte 38
2.1.5.1 Localizar Servidores 39
2.1.5.2 Identificar Serviços Fornecidos por Servidores 39
2.1.5.3 Invocar Serviços num Servidor 40
2.2 Trabalhos Realizados Nesta Área de Conhecimento 42

3 Solução proposta 47
3.1 Introdução 47
3.2 Tecnologias de Suporte Utilizadas 48
3.3 *Broker* Proposto 51
3.3.1 Desenvolvimento 52
3.3.2 Base de Dados do Broker 53
3.4 Os Produtores 58
3.4.1 Tipos de Produtores 58
3.4.1.1 Produtores Reais 58
3.4.1.1.1 Centro de Maquinação 58
3.4.1.1.2 Interface Rs232 59
3.4.1.1.3 Interface Ethernet 60
3.4.1.1.4 Protocolo LSV2 60
3.4.1.2 Produtores Simulados 62
3.4.2 Bases de Dados do Produtor 63
3.4.3 Serviços WEB Propostos 65
3.4.3.1 WSDL dos Produtores 65
3.5 Descrição dos Fluxos de Informação 70
3.5.1 Registo dos Produtores 71
3.5.2 Fase de Pedido de Orçamentos e Negociação 72
3.5.2.1 Localização Produtores 73

3.5.2.2 Pedidos de Orçamentos e Negociação 76
3.5.3 Fase da Realização de Encomendas 78
3.5.4 Seguimento de Encomenda 81

4 RESULTADOS PRÁTICOS 87
4.1 TEMPOS OBTIDOS NA SOLUÇÃO TRADICIONAL 88
4.2 TEMPOS OBTIDOS NA SOLUÇÃO PROPOSTA 94

5 CONSIDERAÇÕES FINAIS, CONCLUSÕES E TRABALHO FUTURO 99
5.1 CONSIDERAÇÕES FINAIS 99
5.2 CONCLUSÕES 101
5.3 TRABALHOS FUTUROS 104

6 BIBLIOGRAFIA 109

7 APÊNDICES 119

Índice de Figuras

Figura 1-1 – Solução proposta 7
Figura 2-1 – Exemplo de um sistema computacional distribuído 13
Figura 2-2 – Camada '*middleware'* num sistema distribuído [Fraunhofer 2006] 14
Figura 2-3 – Arquitectura do DCE [DCE 2005b] 16
Figura 2-4 – Rede computacional distribuída (CORBA) 20
Figura 2-5 – Pedido de um objecto de um cliente CORBA 21
Figura 2-6 – Modelo genérico de um WEB Service 23
Figura 2-7 – Ciclo de vida de um Serviço WEB 25
Figura 2-8 – Estrutura da mensagem SOAP 27
Figura 2-9 – Exemplo de uma mensagem SOAP 28
Figura 2-10 – Estrutura de um documento WSDL 29
Figura 2-11 – Pilha protocolar do UDDI 31
Figura 2-12 – Categorias de informações do UDDI 32
Figura 2-13 – XML Schema do UDDI 33
Figura 2-14 – Estrutura '*businessEntity'* do UDDI 33
Figura 2-15 – Estrutura '*businessService'*do UDDI 35
Figura 2-16 – Estrutura do *'bindingTemplate'* do UDDI 35
Figura 2-17 – Estrutura do *'tModel'* do UDDI 36
Figura 2-18 – Esquema de relacionamentos entre as estruturas do UDDI 37
Figura 3-1 – Solução proposta 47
Figura 3-2 – Tecnologias de suporte utilizadas 50
Figura 3-3 – Relacionamento das várias bases de dados 53
Figura 3-4 – Recurso disponibilizado pelo *Produtor Real* 59
Figura 3-5 – Disponibilização do CN na WEB 60
Figura 3-6 – Protocolo LSV2 61
Figura 3-7 – Base de Dados do *Produtor* 64
Figura 3-8 – Esquema temporal 70
Figura 3-9 – Registo *Produtores* no Servidor UDDI 71
Figura 3-10 – Esquema temporal do registo UDDI 72
Figura 3-11 – Operações do *Broker* na fase de Pedido de Orçamentos e Negociação 72
Figura 3-12 – Pedido de Orçamentos 74
Figura 3-13 – Localização de *Produtores* 74
Figura 3-14 – Localização dos URL dos *Produtores* (WSDL) 75
Figura 3-15 – Exemplo da base de dados do *Broker* com as localizações dos *Produtores* 75
Figura 3-16 – Esquema temporal da localização de *Produtores* 76
Figura 3-17 – Pedido de orçamento do *Broker* aos *Produtores* 76
Figura 3-18 – Esquema temporal da negociação de orçamentos 77
Figura 3-19 – Exemplo do Browser WEB com os orçamentos apresentados ao *Cliente* 77
Figura 3-20 – Operações do *Broker* na fase da Encomenda 78
Figura 3-21 – *Cliente* encomenda um produto ao *Broker* 79
Figura 3-22 – Registo da encomenda no *Broker* 79
Figura 3-23 – *Broker* encomenda um produto ao *Produtor* 80
Figura 3-24 – Esquema temporal da formalização da encomenda 80
Figura 3-25 – Operações do *Broker* na fase de *Seguimento de Encomendas* 81
Figura 3-26 – Seguimento da produção e envio da encomenda 82
Figura 3-27 – Esquema temporal do Seguimento de Encomendas 82
Figura 3-28 – Exemplo de uma página WEB do Seguimento de Encomendas 83
Figura 4-1 – Pedido de Orçamento ao *Produtor* 1 89
Figura 4-2 – Pedido de Orçamento ao *Produtor* 2 90
Figura 4-3 – Peça a produzir ou a adquirir 91
Figura 4-4 – Resposta de Orçamento do *Produtor* 1 92
Figura 4-5 – Resposta de Orçamento do *Produtor* 2 93

Índice de Tabelas

Tabela 1 – Funções do UDDI na fase de Publicação 37
Tabela 2 – Funções do UDDI na fase de Invocação 38
Tabela 3 – Comparação entre as tecnologias de suporte 41
Tabela 4 – Resumo do processo de pedido de orçamentos 91

Lista de Acrónimos

ANACOM	Autoridade Nacional de Comunicações
API	Application Programming Interfaces
ASCII	American Standard Code for Information Interchange
ASP	Active Server Pages
CDS	Cell Directory Service
NC	Numerical Control
CORBA	Common Object Request Broker Architecture
DCE	Distributed Computing Environment
DCOM	Distributed Component Object Model
DDE	Dynamic Data Exchange
DLL	Dynamic Link Library
DNS	Domain Name System
DTS	Distributed File System
GDS	Global Directory Service
GMT	Greenwich Mean Time
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IDL	Interface Definition Language
IIS	Internet Information Services
IP	Internet Protocol
IR	Interface Repository
JPEG	Joint Photographic Experts Group
MIME	Multipurpose Internet Mail Extensions
ODBC	Open DataBase Connectivity
OMG	Object Management Group
OPC	Open Process Control
ORB	Object Request Broker
OSF	Open Systems Foundation
PHP	Personal Home Page
RPC	Remote Procedure Call
SPI	Sociedade Portuguesa de Inovação
SOAP	Simple Object Access Protocol
TCP	Transmission Control Protocol
UDDI	Universal Description, Discovery and Integration
UUID	Universally Unique Identifier
UML	Unified Modelling Language
URI	Uniform Resource Identifier

URL	Universal Resource Locator
XML	Extensible Markup Language
W3C	WorldWide WEB Consortium
WAP	Wireless Application Protocol
WS	Web Services
WSDL	Web Services Description Language

Definições

Ao longo deste estudo são utilizadas algumas palavras-chave: umas que se inserem num contexto mais geral (formatadas a bold e em itálico) e outras num contexto mais objectivo (iniciadas sempre com letra maiúscula e em itálico) e que se enquadram com mais relevância na solução proposta descrita nos capítulos 1 e 3.

Das palavras-chave destacam-se:

Broker – aplicação informática, residente num endereço URL, que disponibiliza determinados serviços capazes de descobrir *Produtores* que podem fornecer um determinado produto, negociar as melhores propostas, realizar encomendas e fazer os seus seguimentos.

Clientes – entidades que acedem ao endereço URL do *Broker* para adquirirem um determinado produto.

Produtos – bens ou objectos disponibilizados pelos *Produtores* ou pelo *Broker*.

Produtores – entidades que, através dos seus recursos e tecnologias, transformam a matéria-prima em produtos solicitados pelos *Clientes*.

Serviços WEB – funções disponibilizadas pelos *Produtores* que permitem a interacção com as restantes entidades envolvidas (*Broker*, *Cliente*).

As palavras-chave de contexto mais geral são as que fazem parte de normas e especificações, de carácter universal ou cuja tradução do inglês para o português não se achou adequada por perderem todo o seu significado ou semântica. Ainda assim será feita, sempre que possível, a tradução mais adequada e correcta.

Exemplo: “skeleton”, ”stub”, “namespace”

O nome das funções ou mensagens propostas no capítulo 3.5 será escrito entre plicas e em itálico, como se pode ver no exemplo que se segue:

‘Pedido de Orçamento’, ‘Encomenda’, ‘Seguimento de Encomenda’.

São ainda formatadas a itálico todas as citações de outros autores, assim como textos transcritos de outros trabalhos ou *sites* da Internet.

Também são utilizados neste trabalho os chamados acrónimos que, para além de se encontrarem listados no início desta dissertação, são escritos ao em letra maiúscula seguidos, na sua primeira utilização, da descrição do acrónimo entre parênteses.

Exemplo: HTML (HyperText Transfer Protocol)

Ao longo deste trabalho são usados dois termos (WEB Services e Serviços WEB) que poderão causar alguma dificuldade na compreensão aquando da sua utilização. Apesar de parecerem semelhantes têm um significado e semântica distintos. O termo WEB Services pretende referir-se à tecnologia de suporte utilizada, enquanto que Serviços WEB pretendem referir-se aos serviços propostos pelo Produtor e que o Broker poderá invocar, recorrendo ou não, aos WEB Services.

"Não existem palavras certas,
Apenas momentos certos para as palavras."

Capítulo 1

Introdução

1 INTRODUÇÃO

A existência de um sistema informático capaz de disponibilizar, partilhar e difundir, num dado momento e num dado lugar, uma determinada informação foi, é e sempre será um factor determinante na diferenciação entre pessoas, entre organizações, entre empresas e entre países. A posse privilegiada do acesso e troca de informações de uma forma rápida e segura constitui uma vantagem e um dos pilares essenciais para o desenvolvimento, em qualquer das áreas científico-tecnológicas, de uma dada empresa ou nação.

O aparecimento do computador e o desenvolvimento dos sistemas informáticos tiveram um papel importante no desenvolvimento tecnológico e científico do século XX. Passou a ser possível a cada utilizador destes computadores instalar as suas próprias aplicações e gerir os seus próprios dados. Mais tarde, passou a ser obrigatório e imperativo a partilha desses mesmos dados com outros utilizadores através de bases de dados remotas localizadas em servidores centrais. Os computadores executavam as aplicações necessárias para aceder ou modificar ou ainda manipular os dados pessoais que apenas serviam a um utilizador em particular. O simples uso do computador para fins pessoais foi-se transformando, de forma rápida e eufórica, numa revolução tecnológica que até aos dias de hoje não sofreu nenhum retrocesso.

Hodiernamente, pode-se dizer que todas as empresas se encontram informatizadas, isto é, têm computadores, redes internas, ligações à Internet e utilizam inúmeras e variadas aplicações informáticas com o objectivo de divulgarem e disponibilizarem os seus produtos e/ou serviços na maior "auto-estrada" da informação. A Internet, através do chamado comércio electrónico, permite que estas empresas assegurem um lugar competitivo no mercado onde actuam. Existem muitas definições para comércio electrónico, das quais se destacam:

Segundo a Wikipédia "Comércio electrónico ou **e-commerce**, ou ainda comércio virtual, é um tipo de transacção comercial feita especialmente através de um equipamento electrónico, como por exemplo, um computador". [Wikipédia 2006]

Segundo a ANACOM (Autoridade Nacional de Comunicações) "A disponibilização da tecnologia World Wide Web, durante os anos 90, permitiu à grande parte do tecido empresarial reequacionar as estratégias de actuação no mercado, provocando alterações profundas no ambiente negocial tradicional, designadamente no modo de relacionamento entre clientes e fornecedores. Esta alteração deu origem a uma nova forma de vender e comprar – o Comércio Electrónico – que se tem convertido num factor fundamental de competitividade e num fortíssimo indutor de produtividade para a generalidade das empresas". [ANACOM 2006]

Ainda segundo a SPI (Sociedade Portuguesa de Inovação) “Em termos simplistas, Comércio Electrónico poder-se-ia definir como “realizar negócios electronicamente”. Uma definição mais elaborada poderia resumir-se a "qualquer tipo de transacção comercial, em que as partes envolvidas interajam electronicamente e não através de trocas ou contactos físicos"”. [SPI 2000]

1.1 O Problema

O comércio electrónico pode ser descrito como a actividade de compra e venda de bens ou serviços apoiados, de alguma forma, por uma infra-estrutura tecnológica computacional distribuída. Com o aparecimento e utilização da Internet, este conceito começou a ser mais utilizado, aceite e divulgado, substituindo ou tornando-se numa alternativa às tecnologias anteriormente existentes (telefone, fax, minitel, cartas, etc.). Com o crescente número de pessoas e empresas a utilizar a Internet para este tipo de negócio, também esta teve necessidade de se adaptar, sofrendo constantes evoluções de forma a dar resposta às necessidades do comércio electrónico. O aumento das velocidades de acesso e a possibilidade de fazer pagamentos através da WEB, tornou-a num sistema ainda mais aberto e propício à sua utilização e expansão. Assim, a WEB afirma-se cada vez mais como um meio rápido e eficaz de disponibilizar, adquirir e trocar informações em qualquer momento e em qualquer parte do mundo.

Cada vez mais as empresas apostam neste meio de informação para se darem a conhecer, assim como aos seus produtos, com o intuito de aumentarem os seus volumes de negócios, tornando-se mais competitivas nos mercados onde se inserem. Mas a competitividade destas empresas não depende apenas do facto de serem, ou não, conhecidas por outras empresas ou pessoas. A maior ou menor rapidez e facilidade em descobrir e/ou subcontratar bens ou serviços a outras empresas, pode ser um meio de reduzir substancialmente os custos de aquisição. Englobados nestes custos, encontram-se os custos de localização e negociação de serviços que são seguramente um dos economicamente mais penalizantes. Como “tempo é dinheiro”, seria desejável reduzir ao máximo os custos de localização e negociação. O tempo despendido na pesquisa de empresas, na redacção de documentos e pedidos de orçamentos, na escrita de propostas, na formalização de encomendas, nas ordens para os bancos, nas confirmações e demais procedimentos burocráticos e financeiros são factores que contribuem significativamente para estes custos.

Um outro factor, não menos importante que o anterior e que tem um peso significativo na competitividade de uma empresa, é a disponibilidade dos serviços. Poder saber a qualquer dia,

a qualquer hora e a qualquer instante o custo e a disponibilidade de um serviço, pode levar a uma considerável redução de tempo e dinheiro. Desta forma, pode-se explorar esta funcionalidade contribuindo para uma redução dos custos de contratação de um serviço.

1.2 Objectivos

Este trabalho tem como objectivos ajudar a reduzir o tempo e/ou os custos da subcontratação (Descoberta de *Produtores*, Orçamentação, Negociação, Realização e Acompanhamento de Encomendas) recorrendo a tecnologias da informação.

Idealmente seria desejável recorrer a uma entidade que pudesse de uma forma simples, rápida, automática e autónoma descobrir o maior número de empresas na WEB, contactá-las individualmente pedindo-lhes os orçamentos para a aquisição de um determinado produto ou serviço. Posteriormente analisar todas as propostas recebidas e no final apresentar a melhor proposta (prazo de entrega mais reduzido e/ou a solução mais económica). Além disso, através dessa entidade deveria ser possível realizar e formalizar automaticamente todo o processo de encomenda desse bem ou serviço e fornecer, a cada instante, a informação relativa ao seu estado (em fila de espera, em produção ou concluída).

No final destas operações, a empresa contratante que pretendesse adquirir um determinado produto ou serviço, não deveria colocar nenhuma das seguintes questões:

- Terão sido contactadas todas as empresas capazes de fornecer uma proposta de orçamento para a aquisição deste bem ou serviço?

- Terei escrito e realizado o pedido de orçamento ou de encomenda, para o produto que pretendo adquirir, de uma forma correcta?

- Já terei recebido as propostas de orçamentos de todas as empresas contactadas?

- Será que analisei coerentemente e correctamente todas as propostas recebidas de forma a escolher a mais vantajosa?

1.3 Situação Actual

As empresas (*Clientes*) que subcontratam serviços ou bens a outras empresas ou fornecedores recorrem frequentemente a uma lista conhecida onde se encontram as localizações e contactos de todos os fornecedores que disponibilizam os serviços que usualmente são contratados. Normalmente, esta lista é limitada a um número reduzido de empresas (*Produtores*) e adaptada às necessidades da empresa contratante (*Cliente*). Quando as empresas pretendem contratar novos serviços (*Produtos*) podem surgir dificuldades de negociação traduzindo-se normalmente em orçamentos economicamente desfavoráveis.
É também possível a um *Cliente,* através da WEB, descobrir informações sobre os fornecedores a partir de palavras-chave, utilizando motores de busca tais como GOOGLE, YAHOO, entre outros. Com base nos resultados da pesquisa, o *Cliente* pode aceder às páginas WEB das várias empresas (*Produtores*), uma de cada vez, e aí pesquisar os *Produtos* que estas disponibilizam. De acordo com as informações disponíveis pode elaborar documentos, faxes e e-mail's para pedir orçamentos a essas empresas. Ao receber os orçamentos das várias empresas (o que pode demorar vários dias, semanas ou meses até obter resposta de todas as empresas contactadas), o *Cliente* tem que, manualmente, compará-los e ordená-los com base nos custos e/ou tempos de entrega. De seguida, tem que elaborar os documentos de adjudicação da encomenda ao *Produtor* pretendido e, posteriormente tem que esperar até a receber nas suas instalações o *Produto* final para poder verificar a sua qualidade. Durante este tempo não é possível ao *Cliente* saber o estado e a evolução da sua encomenda no *Produtor* (se se encontra em fila de espera, em produção ou concluída).
Apesar destas tecnologias e de todos os meios humanos necessários, o tempo gasto na pesquisa de *Produtores*, na elaboração de documentos para pedir orçamentos, formalizar encomendas e realizar transferências bancárias implica custos elevados. Para além disso, o resultado da pesquisa é limitado a um número reduzido de *Produtores*.

1.4 Solução Proposta

Este trabalho propõe Serviços WEB que permitem descobrir e subcontratar bens ou serviços em todo o mundo através da WEB. Também permitem localizar, orçamentar, negociar, encomendar e seguir o estado de produção de uma encomenda de forma automática e autónoma. Para isso, recorrer-se-á a um motor de busca que implementará estes Serviços WEB e a que chamaremos daqui em diante de *Broker*.

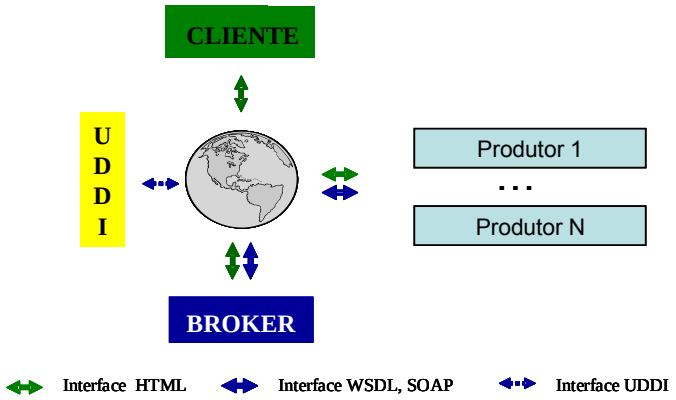


Figura 1-1 – Solução proposta

A solução proposta, ilustrada na Figura 1-1, prevê a existência de um *Broker* capaz de localizar e negociar de forma automática um conjunto de bens ou serviços (*Produtos*), com uma ou mais empresas (*Produtores*), através da sua plataforma WEB. Tanto o *Broker* como a Plataforma WEB de cada *Produtor* implementam os Serviços WEB propostos neste trabalho. Desta forma, um *Cliente* que pretenda adquirir um determinado *Produto*, poderá aceder ao site do *Broker* e aí escolher o *Produto* pretendido, as quantidades e as condições de seriação (preço ou prazo de entrega). Com base nestes requisitos, o *Broker* efectuará um número suficiente de acessos aos vários *Produtores* e no final apresentará ao *Cliente* a melhor proposta para a aquisição desse *Produto*. O *Cliente* poderá também efectuar a encomenda através deste *Broker* e acompanhar a evolução do seu estado, desde o início da produção até à expedição do *Produto* para o *Cliente*. Desta forma, o *Cliente* não necessitará de saber quem é o *Produtor* para cada uma das suas encomendas, negociando e interagindo apenas com o *Broker*.

Mais especificamente, este estudo:

- - Apresenta uma arquitectura conceptual que propõe a existência de um *Broker* e de uma plataforma *WEB* para cada um dos *Produtores*.

- - Propõe Serviços WEB, descritos na linguagem WSDL, que permitem alcançar os objectivos enunciados e que são implementados pelo *Broker* e pela *Plataforma WEB* dos *Produtores*.

- - Apresenta uma implementação possível desse *Broker* e da plataforma WEB dos *Produtores* para avaliar a arquitectura conceptual, os serviços propostos e os seus desempenhos.

Os serviços WEB propostos nesta dissertação, são baseados nas especificações SOAP (Simple Object Access Protocol), WSDL (Web Service Description Language) e UDDI (Universal Description Discovery and Integration) e o protocolo HTTP (HiperText Transfer Protocol):

HTTP – protocolo de comunicação que permitirá a transferência de informação pela WEB entre servidores WEB e browsers a nível mundial.

XML (Extensible Markup Language) – linguagem que permitirá descrever, armazenar, trocar e manipular estruturas de dados, facilitando a sua interpretação assim como a criação de novas aplicações de manipulação e visualização dos dados através da WEB.

SOAP – especificação para o intercâmbio de mensagens entre programas informáticos. Neste caso em particular, é a especificação responsável pela invocação de aplicações remotas e mecanismos de transporte via HTTP, através de RPC (Remote Procedure Call) ou trocas de mensagens (baseadas em XML), em sistemas onde a plataforma ou linguagem de programação não se coloca como parâmetro importante, garantindo desta forma a interoperabilidade e comunicação entre sistemas distribuídos.

WSDL – especificação, baseada no XML, utilizada para descrever Serviços WEB. Neste caso em particular será a linguagem que permitirá a cada *Produtor,* neste contexto, descrever de forma detalhada os Serviços WEB que disponibiliza, as operações que pode realizar, o formato das mensagens a processar, os protocolos que suporta e a sua localização.

UDDI – local na WEB onde se encontram registados Serviços WEB. Neste caso em particular, é a especificação que permitirá ao *Broker* proposto descobrir a nível mundial os *Produtores* capazes de satisfazer uma determinada encomenda. É composto por um conjunto de registos e fornece informações sobre negócios ou entidades aí registadas.

1.5 Organização da Dissertação

No capítulo 2 são descritas as tecnologias e especificações existentes e a forma como elas podem ser aplicadas no âmbito desta dissertação. São ainda referidas as soluções encontradas noutras teses e trabalhos realizados.

No capítulo 3 é apresentada a solução proposta e sua implementação. São também apresentadas as entidades envolvidas, a forma como foram implementadas e a forma como

interagem entre si, bem como as mensagens SOAP e os serviços propostos definidos na linguagem WSDL.

No capítulo 4 são apresentados alguns resultados comparativos do uso da solução proposta em relação à solução tradicional de pedido de orçamentos e encomendas.

No capítulo 5 são apresentadas as conclusões e considerações finais e a possibilidade de trabalhos futuros.

No final desta dissertação é apresentada a bibliografia utilizada no decurso deste trabalho bem como os endereços de páginas WEB consultadas.

São ainda incluídos alguns anexos que complementam o estudo, implementação e escrita desta dissertação.

Capítulo 2

Estado da Arte

2 ESTADO DA ARTE

2.1 Tecnologias de Suporte

Neste capítulo abordam-se várias tecnologias de suporte aos sistemas distribuídos e a forma como elas podem ser utilizadas para suportar a solução proposta nesta dissertação. Entende-se por um sistema computacional distribuído uma rede informática onde existem computadores que podem e pretendem partilhar as suas aplicações ou serviços com outros computadores existentes nessa mesma rede. Assim, pode-se dizer que um determinado computador pode comportar-se, por um lado, como cliente quando requer serviços existentes noutro computador da rede e, por outro lado, como servidor quando fornece os serviços requeridos pelos outros computadores (Figura 2-1). Neste contexto a palavra distribuído significa ainda que estes computadores e outros recursos da rede se encontram geograficamente descentralizados.

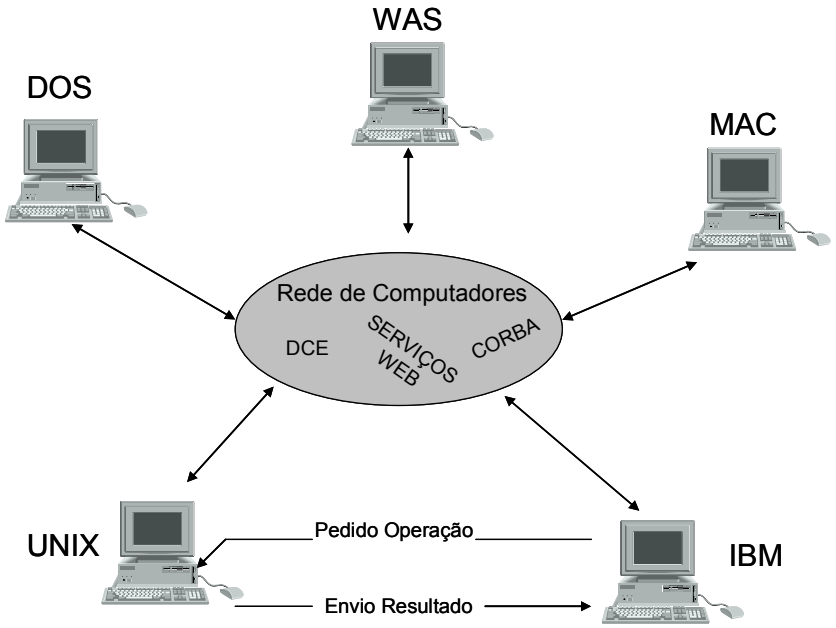


Figura 2-1 – Exemplo de um sistema computacional distribuído

Existem muitas motivações que justificam a utilização de sistemas computacionais distribuídos em alternativa aos sistemas computacionais totalmente centralizados. A utilização de sistemas computacionais distribuídos exige maiores cuidados com a segurança, confiança e desempenho da rede de comunicação. Não obstante, a utilização destes sistemas permite a partilha de recursos e informações entre subsistemas, o que se pode traduzir em significativos ganhos económicos e tecnológicos. [Tanenbaum 1992]

Os programadores ao criarem aplicações distribuídas, pretendem que estas possam interagir com outras existentes noutros computadores, sem necessidade de se realizarem muitas alterações de base ou até mesmo refazê-las completamente. É de todo conveniente aos programadores de aplicações para sistemas distribuídos utilizar especificações, normas e/ou até mesmo protocolos (Tecnologias de Suporte) de forma a permitirem uma mais rápida e transparente implementação e interacção com outras aplicações já existentes na rede distribuída. Estas tecnologias de suporte devem ser capazes de localizar servidores, identificar os seus serviços e fornecer mecanismos de interacção com eles.

Na última década, diversas tecnologias de suporte foram desenvolvidas com o propósito de facilitar o desenvolvimento de aplicações, abstraindo os detalhes da comunicação, invocação de procedimentos e serviços, nomes e localizações. O termo ***'middleware'*** pode ser apresentado como sendo a camada intermédia de software que possui os mecanismos que possibilitam a comunicação entre aplicações distribuídas, diminuindo a complexidade e a heterogeneidade dos diversos sistemas existentes (Figura 2-2).

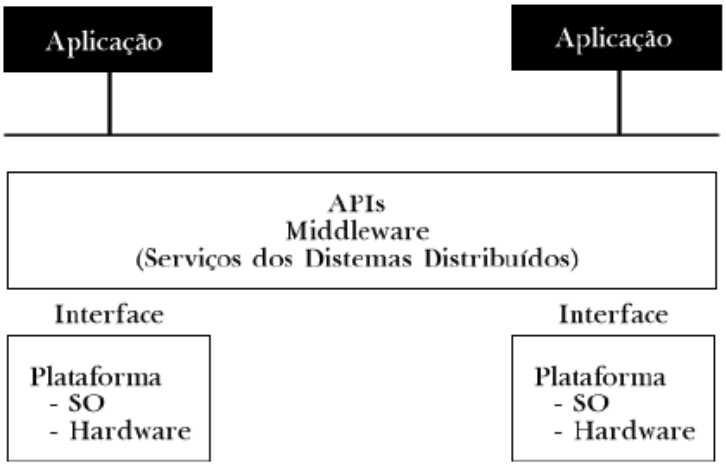


Figura 2-2 – Camada '*middleware'* num sistema distribuído [Fraunhofer 2006]

Tanto o DCE (Distributed Computing Environment), como o CORBA (Common Object Request Broker), o OPC (Open Process Control), como os Serviços WEB (os quais se aprofundarão nas secções seguintes) são tecnologias ***'middleware'*** que suportam a construção e implementação cliente/servidor em sistemas computacionais distribuídos. Ao ter disponível e ao utilizar-se o ***'middleware'*** verifica-se uma considerável redução da complexidade no desenvolvimento de novos sistemas, pois toda a componente de interoperacionalidade entre sistemas operativos, aplicações e plataformas computacionais encontram-se já disponíveis nestes ***'middlewares'***.

2.1.1 DCE

O DCE propõe um conjunto de serviços que pretende facilitar o desenvolvimento, a flexibilidade e a operacionalidade de aplicações informáticas em sistemas distribuídos. Várias implementações deste ambiente distribuído estão disponíveis sob a forma de bibliotecas de funções, podendo ser reutilizadas durante o desenvolvimento de novas aplicações informáticas distribuídas. O DCE é considerado um sistema flexível porque a sua implementação é independente tanto da rede informática como do sistema operativo onde reside. É também usado na construção e integração de aplicações cliente/servidor e fornece os serviços que simplificam o uso ou a chamada de serviços que podem estar fisicamente localizados noutro computador. Tende a ocultar a complexidade inerente ao processamento dos sistemas distribuídos tornando transparente e de fácil compreensão a construção, execução e manutenção de sistemas distribuídos. [Lockhart 1994]

2.1.1.1 Evolução Histórica

No ano de 1990, o DCE v1.0 começou a ser desenvolvido pela OSF (Open Systems Foundation), consórcio formado por mais de 200 empresas donde se destacam a IBM, Hewlett-Packard, Entegrity, Penn State e Chalmers. No ano seguinte saiu a primeira versão do DCE v1.0. Em 1994 foi disponibilizado o DCE v1.1 sob a licença da OSF resultando no FreeDCE. Em Março de 1996 foi apresentado o DCE1.2 e passados quatro anos realizou-se a primeira revisão, resultando daí o DCE 1.2.1. No dia 12 de Janeiro de 2005 foi lançada a última versão do DCE, na sua versão DCE1.2.2 pelo grupo “The Open Group”, apresentando uma licença de software gratuito. [DCE 1996, DCE 2001, DCE 2005a]

2.1.1.2 Arquitectura

A arquitectura do DCE prevê a existência de um conjunto alargado de serviços organizados em vários componentes independentes que interagem entre si, tornando-o numa ferramenta poderosa. [DCE 2005b] Os seus componentes podem ser vistos na Figura 2-3.

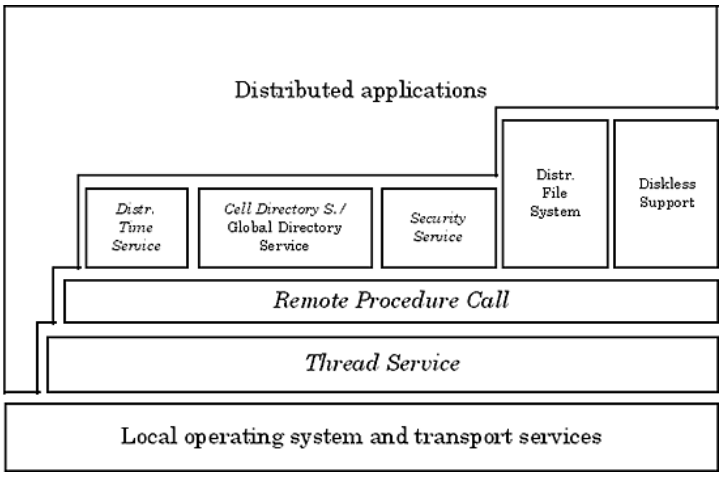


Figura 2-3 – Arquitectura do DCE [DCE 2005b]

O DCE engloba um Serviço de RPC chamado DCE/RPC, um Serviço de Directórios (DCE Directory Service), um Serviço de Segurança (DCE Security System, baseado no Kerberos), um Serviço de Tempo Distribuído (DCE Distributed Time Service) e um Sistema de Ficheiros Distribuídos (DTS – Distributed File System).

DCE/RPC – é o componente principal do DCE, tornando o desenvolvimento de aplicações distribuídas mais simples, fazendo uso de dois processos distribuídos: a subrotina e o método para a invocar. O primeiro encontra-se geralmente implementado no servidor, enquanto que o segundo é usado no cliente. Os parâmetros inerentes à subrotina representam os dados que se pretendem passar quando o cliente questiona o servidor ou durante a resposta do servidor ao cliente.

O DCE/RPC fornece ao programador três serviços básicos:

- O meio de comunicação entre um cliente e um servidor.
- O mecanismo que permite ao cliente localizar o servidor numa rede.
- A transmissão dos dados na rede, convertendo-os nos formatos que achar conveniente.

Serviço de Directórios DCE – este serviço, idealizado para ser utilizado em sistemas distribuídos, fornece um mecanismo fiável onde as aplicações associam informações a nomes. O seu principal objectivo é permitir aos clientes localizar os servidores. No entanto, este serviço pode ser usado pelas aplicações que necessitam de disponibilizar o seu nome e os seus

atributos na rede (aplicações que associam os seus nomes aos respectivos endereços, números de telefone, etc.).

O Serviço de Directórios DCE é composto por dois componentes: o GDS (Global Directory Service) e o CDS (Cell Directory Service). O CDS fornece a identificação do nó onde se inicia uma ramificação terminal chamada ***'Cell'***. O GDS fornece a identificação da ramificação principal onde constam todos os CDS, isto é, todos os CDS aparecem como ramificações do GDS. O GDS está de acordo com a norma X.500 [X500 2006] no que respeita à identificação de serviços, ou seja, todos os sistemas que utilizarem o DCE como base podem ligar-se a outros sistemas exteriores, públicos ou privados, para aceder a serviços sem necessidade de recorrerem a alterações de software.

A WEB usa o protocolo de identificação DNS (Domain Name Service). No DCE, o CDS pode usar o DNS como uma directoria global, substituindo-o ao GDS, permitindo aos utilizadores do DCE descobrir e interagir com outras redes existentes na WEB. Cada utilizador apenas vê uma única árvore com várias ramificações onde constam os nomes ou ***'namespace'*** das aplicações disponibilizadas nas várias redes. Sendo um ***'namespace'*** distribuído, o Serviço de Directórios DCE consiste numa base de dados com um grande número de registos, independentes e localizados nos vários computadores que integram os vários nós. Assim, se houver um problema num dos nós do Serviço de Directórios DCE (computador existente na rede), não há impedimento que os restantes utilizadores acedam a outros computadores. O Serviço de Directórios DCE comporta-se como um sistema replicado, isto é, existem nele múltiplas cópias da identificação de um ***'namespace'*** nos vários nós do sistema. A replicação, por um lado ajuda a tornar o sistema mais fiável, já que elimina a possibilidade de falhas num ponto específico da rede e, por outro lado, permite aumentar a performance, visto a informação do ***'namespace'*** poder ser lida num nó mais próximo, reduzindo o tráfego de informação na rede. Para além destas vantagens, a replicação implica uma constante actualização das bases de dados, sendo ainda assim benéfica já que o número de leituras às bases de dados é largamente superior ao número de actualizações necessárias.

Serviço de Segurança DCE – este serviço fornece os mecanismos necessários para assegurar que os serviços existentes na rede estejam apenas disponíveis para utilizadores autorizados. Este problema não se coloca caso os utilizadores se encontrem num computador local, dado que os sistemas de segurança são assegurados localmente. O problema de segurança coloca-se somente nos sistemas ou redes distribuídas e pode ser descrito em três partes:

- Ao estabelecer uma ligação com um determinado utilizador, algures numa localização da rede, tem que se ter a certeza que essa pessoa é a certa (Autenticação).

- É necessário determinar quais os privilégios associados a cada utilizador (Autorização).

- É necessário verificar se esses privilégios permitem a realização das operações desejadas pelos utilizadores (Controlo de Acessos).

Um ponto obrigatório para todos os utilizadores de sistemas distribuídos é o seu registo na rede e não num computador específico. Isto impossibilita que um utilizador não autorizado consiga entrar na rede, tenha acesso às informações que lá circulam e consiga ler informações confidenciais, palavras-chave ou até mesmo, enviar mensagens em nome de um utilizador autorizado para essa rede. Portanto, um sistema distribuído para ser robusto, seguro e fiável não deve permitir o acesso a serviços se o sistema de segurança não estiver activo. O Sistema de Segurança DCE vem resolver todos estes problemas. Tal como o Serviço de Directórios DCE, o Sistema de Segurança DCE é também distribuído e replicado, autentica os utilizadores, determina as suas autorizações, é imune aos ataques que envolvem a captura de mensagens na rede e nunca transmite palavras-chave.

Serviço Tempo Distribuído DCE – este serviço é responsável pela sincronização dos relógios dos vários computadores existentes na rede. Se num dos nós da rede existir um computador remoto oficial que forneça as horas GMT (Greenwich Mean Time**)**, o Serviço Tempo Distribuído DCE actualiza os relógios de todos os computadores com a hora fornecida por esse computador oficial. Esta sincronização dos relógios torna-se um aspecto importante quando o protocolo utilizado pelo Serviço de Segurança assume que todos os nós da rede têm a mesma hora. Existem sistemas de segurança que fornecem uma licença cuja duração caduca com o tempo de utilização. Existem ainda programas do tipo “cópias de segurança” que são controlados e lançados pela data de criação dos ficheiros a salvaguardar. Se houver uma discrepância nos relógios dos vários computadores, pode dar-se o caso destas licenças expirarem prematuramente ou de o processo criação de cópias de segurança não ser realizado correctamente.

Sistema Distribuído de Ficheiros – este sistema fornece os mecanismos para as aplicações clientes poderem aceder a ficheiros localizados no servidor, como se estivessem localizados localmente. Todos os nós na rede identificam o mesmo ficheiro pelo mesmo nome e vêem-no localizado no mesmo endereço. O Sistema Distribuído de Ficheiros também pode utilizar a

replicação de dados nos diferentes nós, mantendo desta forma a consistência entre as cópias. O número e a localização das cópias são controlados pelo administrador do sistema. Este sistema é implementado no topo da pirâmide dos serviços DCE, como se pode ver na Figura 2-3 (ver página 13), podendo ser dito que o Sistema Distribuído de Ficheiros possui toda a interoperacionalidade do DCE.

2.1.2 CORBA

O CORBA (Common Object Request Broker Architecture) foi desenvolvido pelo OMG (Object Management Group). [OMG 2006, OMG 2006b] Esta organização congrega mais de 800 empresas e tem como objectivo facilitar o desenvolvimento de aplicações distribuídas. Estão disponíveis no mercado implementações desta arquitectura, sob a forma de bibliotecas de funções que permitem aos programadores desenvolver rapidamente aplicações distribuídas.
O CORBA constitui um mecanismo fiável que permite estabelecer e simplificar a troca de dados entre sistemas distribuídos e heterogéneos. Face à diversidade de hardware e software utilizados pelos programadores no desenvolvimento de sistemas distribuídos, o CORBA é o responsável pela comunicação e interacção destas aplicações com outras existentes dentro da mesma rede, tornando-as totalmente transparentes. Assim, pretende-se que uma aplicação distribuída consiga interagir com outras aplicações, independentemente da linguagem de programação, da sua localização, dos seus serviços e dos sistemas operativos. No CORBA, mais importante do que o método ou a forma como as aplicações são implementadas, é fundamental o modo como os objectos são referenciados e como são acedidos no universo dos restantes objectos existentes na rede. Assim, esta tecnologia disponibiliza transparência face à localização dos objectos, às linguagens e às redes, permitindo integrar aplicações já existentes e reduzir custos de manutenção.
O CORBA é um dos modelos mais populares de objectos distribuídos juntamente com o DCOM (Distributed Component Object Model) da MICROSOFT.

2.1.2.1 História e Evolução

Em Outubro de 1991, o CORBA 1.0 foi apresentado como versão inicial que incluía: o modelo de objectos CORBA, a linguagem da definição de relação IDL (Interface Definition Language) e as relações de programação de aplicações API (Aplication Programming Interfaces) para pedidos de gestão e invocação dinâmicos DII, integrando ainda uma biblioteca para a linguagem de programação C. Em Fevereiro de 1992 foi publicado o CORBA 1.2, a primeira versão da especificação do CORBA, que permitiu corrigir algumas anomalias da especificação

original, acrescentar definições de relação e optimizar a gestão de memória em relação ao modelo original. Em Agosto de 1996, o CORBA 2.0 sofreu a primeira revisão geral, tendo-lhe sido acrescentadas novas funcionalidades. Em Junho de 1999, a versão CORBA 2.3 sofreu novas revisões e foram-lhe adicionadas outras especificações (Portabilidade IDL/Java, Biblioteca de Java para IDL e de IDL para Java, Biblioteca de C++, entre outras). Em Outubro de 2000 foram incluídas novas especificações, surgindo o CORBA 2.4 (CORBA em tempo real, Núcleo e RTF2.4). Em Setembro de 2001 deu-se uma nova revisão surgindo o CORBA 2.5, que incluiu algumas novas especificações (optimização da transmissão de mensagens, CORBA Tempo Real). Em Julho de 2002, o CORBA 3.0 fez o CORBA Mínimo e CORBA em Tempo Real aparecerem como documentos separados (ambos tinham sido acrescentados ao Núcleo CORBA no Lançamento 2.4).

2.1.2.2 Arquitectura

A arquitectura CORBA permite a um cliente aceder a recursos computacionais e informações distribuídas a partir de aplicações correntes (Figura 2-4). Tem como objectivo definir e proporcionar interfaces de programação para os chamados componentes ORB (Object Request Broker). O ORB é um mecanismo básico que permite que aplicações distribuídas comuniquem entre si de uma forma transparente, quer se encontrem localmente na mesma máquina, quer se encontrem algures num computador existente numa rede.

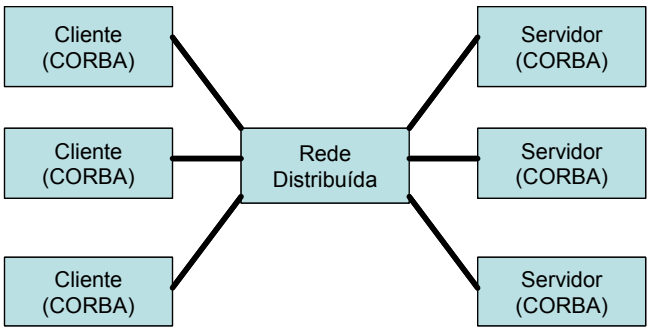


Figura 2-4 – Rede computacional distribuída (CORBA)

Um cliente não necessita saber quais os mecanismos para activar ou comunicar com uma determinada aplicação, nem como esta é implementada, nem mesmo qual a sua localização. O ORB constitui uma base para construir aplicações distribuídas e proporcionar uma inter-operacionalidade entre aplicações tanto em ambientes homogéneos como heterogéneos.

O conceito de objecto local (cliente) e objecto remoto (servidor) também são utilizados na arquitectura CORBA para definir os objectos envolvidos na chamada de um método remoto. Na Figura 2-5 pode-se ver como é feito um pedido de um cliente a um servidor localizado num

sistema distribuído. [Santos 1996] No final deste processo, o cliente tem que saber localizar o servidor, interagir com ele e conseguir chegar ao ***'Object Implementation'*** via ORB. No entanto, a aplicação cliente tem primeiramente que saber o nome do serviço que pretende invocar para o transmitir ao ORB. Este vai pesquisar, numa espécie de base de dados chamada IR (Interface Repository), onde constam os nomes dos serviços ou objectos e as suas localizações na rede, identificando, de forma unívoca, o objecto solicitado pelo cliente. Todo o objecto tem que escrever a sua identificação e localização no IR, visto ser esta a única forma de poder ser identificado e invocado por aplicações distribuídas. O ***'Client Interface'***, objecto do lado do cliente, é definido recorrendo à IDL (Interface Definition Language), que é uma linguagem genérica para descrever interfaces e não uma linguagem de programação. A sintaxe da IDL apenas permite definir os métodos e os seus valores de retorno, não estando nela contemplados os fluxos de controlo. [Vinoski 1997]

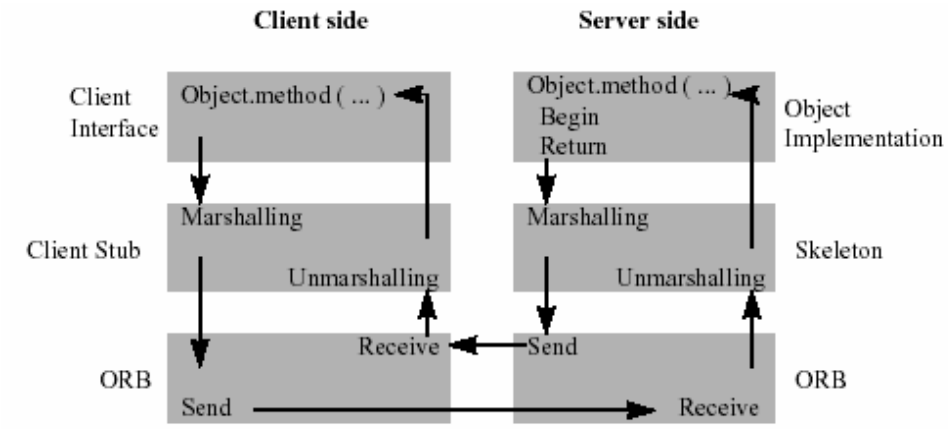


Figura 2-5 – Pedido de um objecto de um cliente CORBA

ORB **–** fornece um mecanismo que permite a um cliente comunicar, de uma forma transparente, com aplicações que se encontram localizadas num sistema distribuído. Também simplifica a programação, a localização das aplicações distribuídas, sendo ainda responsável pela transmissão e recepção das mensagens previamente formatadas pelo componente ***'Stub'***.

'Client Stub' **–** é neste componente do CORBA que ocorre, de uma forma automática e sem intervenção do programador, a formatação dos parâmetros a serem transmitidos pelo ORB para a rede (***'marshlling'***). O ORB ao receber uma mensagem do servidor vai fazer o inverso (***'unmarshlling'***), reconfigurando a mensagem recebida da rede para uma mensagem capaz de ser compreendida pelo cliente.

'***Skeleton***' **–** é este componente do CORBA, localizado no lado do servidor, que vai chamar o objecto pretendido e executar a sua implementação quando recebe um pedido do cliente via ORB. Realiza da mesma forma que o **'*Client Stub'*** as operações de ***'marshling'*** e ***'unmarshling'*** às mensagens transmitidas e recebidas da rede.

2.1.3 OPC

Esta secção apresenta as normas que a fundação OPC (Open Process Control) tem desenvolvido para permitir e facilitar o controlo e a supervisão das instalações fabris. Esta fundação envolve a Microsoft e um grande número de fabricantes e líderes mundiais na área da automação industrial. Ao longo dos anos esta fundação tem vindo a definir um conjunto de normas nomeadamente a OPC-DA (Data Access) e a OPC-XML-DA que serão sucintamente apresentadas nas secções seguintes.

2.1.3.1 OPC – DA (Data Access) Custom Interface Standard (v3.00)

Esta norma define como é possível aceder aos recursos de instalações fabris (Controladores Lógicos Programáveis - PLC, Interfaces Homem Máquina – HMI, etc.) a partir de computadores. Mais concretamente, esta norma permite que aplicações informáticas desenvolvidas em sistemas operativos Windows (OPC Foundation, 2003a) possam aceder com facilidade aos recursos fabris para, por exemplo, escrever ou ler facilmente valores nas suas memórias internas.
Seguindo esta norma, a maior parte dos fabricantes de automação industrial fornecem programas para o ambiente Windows chamados Servidores OPC (***'OPC Servers'***). Estes servidores têm como objectivo estabelecer a comunicação com os recursos fabris e simultaneamente disponibilizar uma interface OPC para as outras aplicações Windows, independentemente da diversidade e da especificidade de cada recurso fabril. Desta forma, um integrador de sistemas pode rapidamente desenvolver aplicações Windows que actuem como clientes OPC, de modo a controlar ou a monitorizar a produção na instalação fabril, sem terem que se preocupar com o desenvolvimento de programas específicos para cada tipo de recurso fabril.

2.1.3.2 OPC-XML-DA

De acordo com esta norma, um servidor OPC-XML-DA pode trocar informação com um cliente OPC-XML-DA através da Internet. Esta especificação define como o cliente pode fazer pedidos ao servidor usando mensagens escritas em XML, encapsuladas em envelopes SOAP e transferidas de acordo com o protocolo HTTP. Esta norma prevê também que o cliente possa recorrer a servidores UDDI para descobrir quais os fornecedores de serviços adequados aos seus pedidos. Tem ainda a grande vantagem de não estar limitada a sistemas operativos

Windows, como era o caso da norma OPC-DA. Os clientes e os servidores OPC-XML-DA podem também ser implementados em sistemas Unix, Linux, entre outros.

2.1.4 WEB Services

A Internet surge cada vez mais como uma plataforma essencial na integração de sistemas e aplicações baseadas num conjunto de protocolos, normas e especificações aceites universalmente, permitindo desta forma, a comunicação entre duas ou mais aplicações que pretendem trocar dados entre si. No entanto, os protocolos TCP (Transmission Control Protocol), IP (Internet Protocol) e HTTP não dispõem de todos os mecanismos necessários para que esta integração seja possível. Foi neste contexto que surgiu a necessidade de utilização de WEB Services, de forma a criar uma interoperabilidade universal utilizando apenas os protocolos, normas e especificações disponibilizados pela WEB.

Os WEB Services são todos os serviços que podem ser disponibilizados através da WEB e que permitem uma troca e processamento automático de dados entre aplicações informáticas. De acordo com a W3C (WorldWide WEB Consortium), os WEB Services são descritos através de documentos WSDL e são evocados remotamente através de mensagens SOAP. É também utilizada a especificação UDDI como mecanismo de registo e pesquisa destes serviços WEB a nível mundial.

Os principais participantes do modelo de arquitectura orientada aos serviços são: o *Cliente*, o fornecedor de Serviços WEB e o servidor UDDI (Figura 2-6). [Ribeiro 2005]

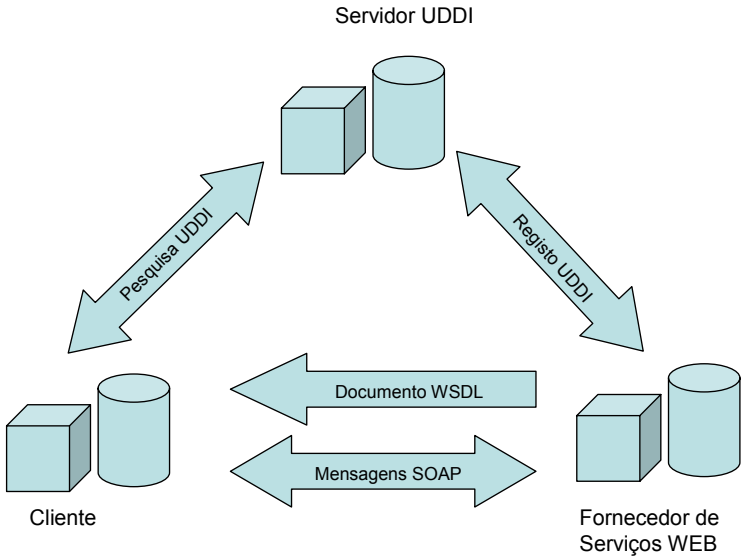


Figura 2-6 – Modelo genérico de um WEB Service

Para além de utilizarem as especificações padrão da WEB (SOAP, WSDL, UDDI, HTTP), estas aplicações podem recorrer a outras plataformas de codificação, transporte, registo e/ou pesquisa de serviços WEB.

Os WEB Services são uma tecnologia emergente, sobre a qual muito se tem escrito. Enquanto que uns a consideram como um caminho a seguir no âmbito do desenvolvimento de aplicações distribuídas, outros apenas vêem nela mais uma evolução de um conceito antigo (CORBA).

Existem várias definições para os WEB Services, como sendo:

Segundo W3C Serviços Web Arch WG: "Um WEB Service é uma aplicação de software identificada por uma URI (Uniform Resource Identifier) cujas interfaces e ligações são capazes de serem definidas, descritas e descobertas por artefactos XML. Suportam interacções directas com outras aplicações de software usando mensagens baseadas e, XML via protocolos baseados na Internet". [W3C 2006]

Segundo Java Technology: "WEB Services são aplicações empresariais baseadas na Web que usam o XML standard e protocolos de comunicação para trocar dados entre si". [Java 2006]

Segundo a Gartner Group " […] componentes de software espalhados que interagem dinamicamente uns com os outros através de tecnologias Internet standard[…]"[Gartner 2006]

Segundo a Forrester Research " […] ligações automáticas entre pessoas, sistemas e aplicações que expõem elementos de funcionalidade de negócio como um serviço de software e criam um novo valor de negócio […] " [Forrester 2006]

O ciclo de vida de um Serviço WEB (Figura 2-7) pode ser descrito em quatro estados distintos: o Registo, a Pesquisa, a Descrição e a Invocação. Estes estados podem ser invocados ou utilizados pelas aplicações que utilizam os Serviços WEB de forma directa ou indirecta. Directamente, acedendo a uma página disponibilizada pelo fornecedor dos Serviços WEB onde é possível registar e pesquisar um determinado serviço. Indirectamente, sendo a aplicação informática do cliente a realizar as operações de registo e pesquisa através de chamadas de funções aquando da sua execução.

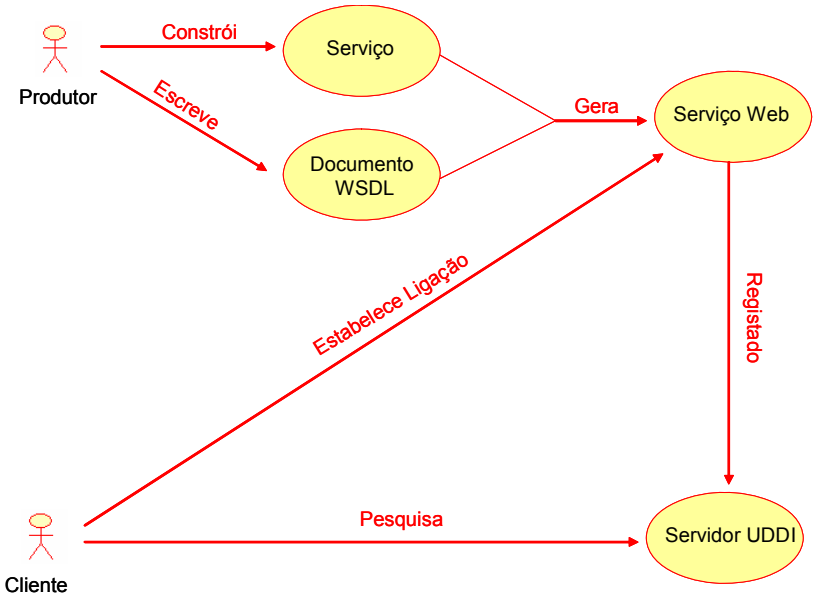


Figura 2-7 – Ciclo de vida de um Serviço WEB

O ciclo de vida de um Serviço WEB contempla quatro etapas principais [Lopes 2004]:

A primeira etapa consiste na construção e registo do Serviço WEB. Este pode ser programado em qualquer linguagem de programação que permita interpretar XML. Nesta fase encontra-se a especificação da interface do serviço, definida através de um documento WSDL (Descrição).

A segunda etapa consiste no registo do serviço WEB num dos servidores UDDI. Nesta fase, são fornecidas aos servidores UDDI, todas as informações respeitantes ao Serviço WEB que se pretende disponibilizar (Registo).

A terceira etapa consiste na pesquisa de um Serviço WEB nos servidores UDDI por parte de um utilizador (aplicação cliente). Nesta etapa deverá ficar a conhecer-se a localização do Serviço WEB pretendido, os seus parâmetros de entrada e saída, o nome dos serviços ou as funções disponibilizadas, etc. (Pesquisa).

Na quarta etapa dá-se o estabelecimento da ligação ou comunicação entre o utilizador (aplicação cliente) e o fornecedor Serviço WEB através de mensagens SOAP/XLM.

Veja-se mais pormenorizadamente cada um dos estados dos Serviços WEB:

Descrição – processo pelo qual o Serviço WEB expõe o seu API (Application Programming Interface) mais especificamente o seu documento WSDL (Web Services Description Language), tendo a aplicação cliente acesso a toda a interface do Serviço WEB. Desta forma, a

aplicação cliente passa a dispor as informações de todas as funcionalidades disponibilizadas pelo Serviço WEB, assim como os métodos, protocolos e tipos de mensagens que permitem aceder às suas funcionalidades. Neste estado é utilizada a norma WSDL.

Registo – processo opcional através do qual o fornecedor do Serviço WEB dá a conhecer a sua existência e a do seu serviço através de um registo num dos servidores UDDI existentes (IBM, MICROSOFT, GOOGLE). Neste estado é utilizada a norma UDDI.

Pesquisa – processo opcional através do qual uma aplicação cliente toma conhecimento da existência do Serviço WEB pretendido, através de uma pesquisa nos servidores UDDI. Neste estado é utilizada a norma UDDI.

Invocação – processo pelo qual o cliente e o servidor interagem através do envio de mensagens de ***"input"*** e de eventuais recepções de mensagens de ***"output"***. Neste estado é utilizada a norma SOAP ou XML.

2.1.4.1 SOAP

As mensagens SOAP são escritas no formato XML e obedecem a uma organização conhecida e bem definida. [SOAP 2003] Devido ao elevado nível de formalismo destas mensagens é possível realizar o seu processamento automático através de aplicações informáticas, o que não era possível com o uso de mensagens HTML.
Este tipo de mensagens é essencialmente unidireccional no sentido emissor/receptor podendo no entanto ser combinadas, de forma a incluir, numa única mensagem, a informação relativa ao pedido e à resposta, tornando-se bidimensionais.

2.1.4.1.1 História e Evolução

No Verão de 1998, a USERLAND SOFTWARE (Dave Winer, Don Box), uma empresa aliada da MICROSOFT, apresentou uma especificação chamada XML-RPC. No ano de 1999, a MICROSOFT lançou a primeira versão do SOAP1.0. Em Maio de 2000, Don Box, DEVELOPMENTOR, IBM e a MICROSOFT apresentaram a versão SOAP1.1 numa nota do W3C. Em Dezembro de 2001 foram publicados os primeiros artigos sobre SOAP1.2. Em 2004, a W3C apresentou a última versão do SOAP (SOAP1.2) como uma especificação. [SOAP 2005a, SOAP 2005b]

2.1.4.1.2 Estrutura das Mensagens SOAP

De forma a perceber a estrutura de uma mensagem SOAP é normal estabelecer a analogia com uma carta de correio. [Lopes 2005] A razão desta analogia reside no facto da estrutura destas mensagens estar dividida em quatro secções, como se pode ver na Figura 2-8.

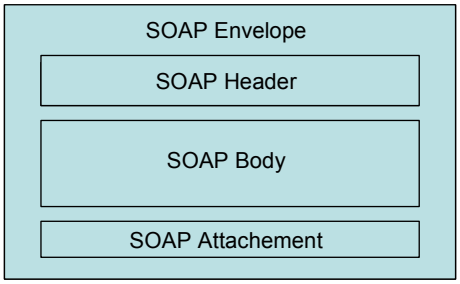


Figura 2-8 – Estrutura da mensagem SOAP

Envelope (*'Envelope'*) – este elemento encontra-se no início de cada mensagem SOAP, identificando-a como tal. O início das mensagens SOAP identifica-se através da etiqueta *"<envelope"* e termina com a etiqueta *"/envelope>".* Tem como ***'namespace'*** obrigatório o endereço http://schemas.xmlsoap.org/soap/envelope e poderá ter um atributo adicional *'**encodingStyle'*** que define a forma como os dados são representados na mensagem.

Cabeçalho (*'Header'*) – este elemento encontra-se dentro do elemento *Envelope* e começa com a etiqueta *"<header",* terminando com a etiqueta *"/header>"*. É um campo opcional que pode conter vários elementos normalmente com informações de controlo e informações que permitem adicionar novas funcionalidades (autenticação, transacções, encriptação, etc.), sem ser necessário alterar as especificações do SOAP. Este elemento, no caso de existir, deverá ser o primeiro sub-elemento do elemento *Envelope*.

Corpo ('Body') – este elemento sempre presente nas mensagens SOAP, contém a mensagem propriamente dita. Insere-se igualmente dentro do elemento *Envelope* e o seu início pode ser identificado através da etiqueta *"<body"* e o seu final através da etiqueta *"/body>"*.

Anexo ("Attachment") – este elemento é opcional e encontra-se definido dentro do elemento *Envelope*. Podem existir vários elementos deste tipo num <envelope>. É normalmente usado para transmitir anexos, recorrendo a mecanismos MIME (Multipurpose Internet Mail Extensions).

Um exemplo de mensagens SOAP pode ser visto na Figura 2-9.

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
    <Header>
       <t:Transaction
            xmlns:t=”some-URI”
            SOAP-ENV:mustUnderstand=”1”>
       </t:Transaction>
   <Header>
   <Body>
       <find_tModel  maxRows="50" xmlns="urn:uddi-org:api_v2" generic="2.0">
        <findQualifiers>
          <findQualifier>sortByDateDesc</findQualifier>
        </findQualifiers>
        <name >%Maquinação UA%</name>
       </find_tModel>
    </Body>
  </Envelope>
```

Figura 2-9 – Exemplo de uma mensagem SOAP

2.1.4.2 WSDL

Um documento WSDL permite descrever Serviços WEB de entidades que pretendam disponibilizar serviços ou produtos na WEB. Este tipo de documento (escrito em XML) permite descrever um Serviço WEB, nomeadamente a forma de o invocar, as operações disponibilizadas pelo produtor, o protocolo de comunicação e o tipo de dados utilizados. [WSDL 2006b, WSDL 2006c, WSDL 2006d]

2.1.4.2.1 História e Evolução.

Em Março de 2001, o WSDL 1.1 foi sugerido numa nota do W3C (pelas empresas ARIBA, IBM e MICOSOFT) como um documento utilizando a especificação XML para descrever serviços WEB. Esta nota também referia como usar o WSDL juntamente com SOAP1.1, HTTP GET/POST e MIME. Em Dezembro de 2001 foram publicados os primeiros artigos sobre o WSDL1.2. Em Julho de 2002 foi criado o primeiro serviço a utilizar o WSDL1.2 e em Agosto de 2005 foi implementada a primeira versão do WSDL2.0. [WSDL 2006a]

2.1.4.2.2 Estrutura dos Documentos WSDL

Um documento WSDL obedece a uma estrutura conhecida e bem definida, onde cada elemento tem uma função clara e objectiva. Esta estrutura, representada na Figura 2-10, assenta num elemento principal chamado *Definitions*, no qual podem existir cinco sub-elementos. [Menéndez 2002]

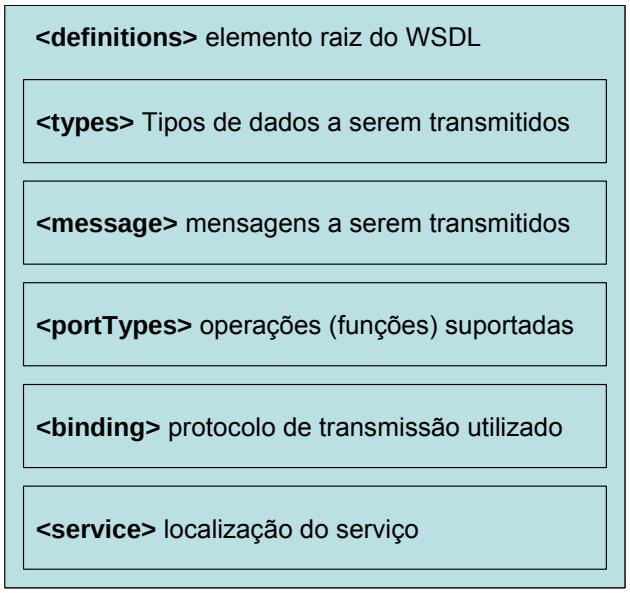


Figura 2-10 – Estrutura de um documento WSDL

***definitions* –** elemento raiz que contém todas as definições respeitantes a todos os elementos presentes no ficheiro WSDL.

***types* –** elemento onde constam as informações que identificam o tipo de dados usados na troca de mensagens.

messages* –** elemento que descreve de uma forma abstracta os fluxos de informação, nomeadamente os nomes dos parâmetros necessários para invocar correctamente cada uma das funções disponibilizadas pelo Serviço WEB ('message parts'***).

portTypes* –** elemento onde são descritas as operações (métodos) disponibilizadas no Serviço WEB. São compostos por um conjunto de mensagens de ***'input' e ***'output'***, podendo ainda ser encontradas mensagens de erro (**'operation'**).

***bindings* –** elemento que identifica o protocolo de transporte a utilizar.

services* –** elemento constituído pelo conjunto de elementos ***'port' que identificam os diferentes pontos de acesso/localização do Serviço WEB a invocar, normalmente através do

seu endereço URL. Encontra-se ainda neste elemento a localização do endereço do Serviço WEB **('Ports'**).

documentation **–** elemento destinado a comentários, podendo conter texto ou outros elementos XML. Pode encontrar-se dentro do elemento ***'definitions'*** ou dentro de qualquer um dos outros elementos do ficheiro WSDL.

2.1.4.3 UDDI

Os Serviços WEB estão a assumir um papel cada vez mais importante no comércio electrónico. As empresas publicam os seus serviços e/ou produtos que podem ser utilizados ou requeridos por outras empresas de forma a executar uma transacção. A gestão de um ambiente onde existem poucas requisições de Serviços WEB pode ser uma tarefa simples. No entanto, com o número crescente de empresas a utilizarem a Internet como meio de divulgação e realização de negócios, o número de trocas de informações entre elas tende a aumentar significativamente. Nesta situação surge um problema:

Como garantir que todas as novas empresas sejam descobertas e contactadas?

Se for realizado um contacto pessoal como garantir que todas as opções sejam tidas em conta?

O UDDI é um directório de serviços onde podem ser registados e pesquisados Serviços WEB. Esta tecnologia não é nada mais do que uma estrutura que permite disponibilizar, utilizar e pesquisar serviços na Internet. O UDDI é uma interface de serviço descrita através de documentos WSDL, que constitui uma parte integrante da plataforma .NET da Microsoft e que comunica utilizando mensagens SOAP transmitidas através do protocolo HTTP. Assim, o UDDI tem como objectivos proporcionar a um fornecedor de serviços WEB ou a um utilizador (cliente), uma ferramenta simples, rápida e automática de:

- Encontrar o melhor serviço entre os milhares que estão dispostos na rede.
- Definir como configurar a comunicação, uma vez que o serviço foi escolhido.
- Permitir, de forma fácil, o acesso a novos serviços, assim como aos já existentes.

A Figura 2-11 mostra a localização do UDDI numa pilha protocolar onde se encontram outras tecnologias e especificações já referidas nas secções anteriores.

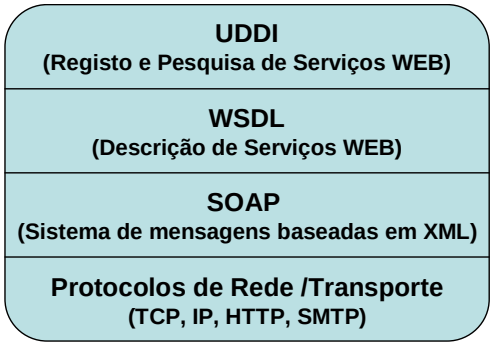


Figura 2-11 – Pilha protocolar do UDDI

2.1.4.3.1 História e Evolução

O UDDI foi iniciado por três empresas: IBM, MICROSOFT e ARIBA. Estas empresas juntaram-se com o intuito de criar um método standard capaz de divulgar e descobrir Serviços WEB, mas que ao mesmo tempo pudesse ser invocado independentemente da linguagem ou plataforma de desenvolvimento. Em Setembro de 2000, o UDDI v1.0 foi publicado de forma a permitir o registo de serviços na WEB. Em Junho de 2001 foi descrita uma especificação sobre a arquitectura que os Serviços WEB deveriam seguir, de forma a permitir uma maior flexibilidade (UDDI v2.0). Em 2002 saiu a especificação UDDI v2.0 através da OASIS. Em Julho de 2003, o UDDI v3.0 foi considerado o suporte de interacção de implementação privada e pública. Em 2004 saiu como especificação UDDI v3.0 através da OASIS. [UDDI 2006]
O UDDI baseia-se nos padrões definidos pelo W3C e IETF (Internet Engineering Task Force), tais como o XML, o HTTP e o DNS. Adicionalmente, as características de programação das plataformas são suportadas, adoptando-se versões do SOAP conhecidas como as especificações de mensagens XML. [UDDI 2004]

2.1.4.3.2 Estrutura do UDDI

O servidor UDDI disponibiliza uma base de dados para armazenar informações sobre Serviços WEB. Estas informações encontram-se replicadas em várias bases de dados, visto existirem várias empresas (MICROSOFT, IBM, SAP, ORACLE, entre outras) a suportarem este tipo de serviço. Desta forma, uma aplicação que pretenda registar um Serviço WEB apenas tem de se registar num único servidor, não havendo necessidade de o fazer em cada um dos vários

servidores UDDI disponíveis. Mas o servidor UDDI não é apenas uma base de dados onde constam as ditas informações, o UDDI disponibiliza ainda três serviços básicos:

Registo – este serviço permite à empresa registar-se e registar o serviço que pretende ver disponibilizado na WEB.

Pesquisa – este serviço permite a uma aplicação cliente obter informações sobre serviços WEB e sobre a empresa.

Mapeamento – este serviço descreve a forma de evocar e interagir com o fornecedor do Serviço WEB pretendido, após a sua localização.

O UDDI utiliza o *"XML Schema"* [XML Schema 2004a, XML Schema 2004b, XML Schema 2004c] para descrever formalmente as suas estruturas de dados. Estas estruturas, resultantes do processo de registo, são bem definidas e permitem realizar uma posterior pesquisa de informações específicas. A pesquisa é feita pelo tipo de serviços, funcionando um pouco como as páginas amarelas dos Serviços WEB. Assim, a informação guardada no directório do UDDI pode ser apresentada em três tipos de categorias: as Páginas Brancas (***'White Pages'***), as Páginas Amarelas (***'Yellow Pages'***) e as Páginas Verdes (***'Green Pages'***). Estas informações encontram-se no directório do UDDI organizadas como se mostra na Figura 2-12.

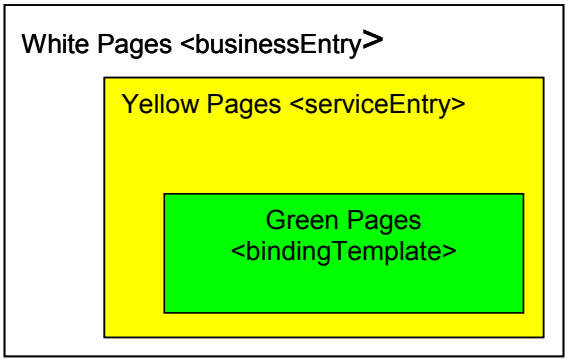


Figura 2-12 – Categorias de informações do UDDI

Páginas Brancas ('White Pages') – estrutura do directório UDDI que contém a identificação básica da empresa que fornece o Serviço WEB (nome, morada, telefone, descrição da empresa, entre outros), podendo haver outros dados alternativos pelos quais a empresa pode ser reconhecida.

Páginas Amarelas ('Yellow Pages') – estrutura do directório UDDI que contém a informação quer do Serviço WEB, quer do seu fornecedor, por categorias em que estes se inserem (tipo de serviço, localização geográfica, etc.).

Páginas Verdes ('Green Pages') **–** estrutura do directório UDDI que contém a informação técnica que permite descrever o comportamento e as funções disponibilizadas pelo serviço (localização do documento WSDL ou endereço URL).

O registo completo de um Serviço WEB no UDDI consiste no preenchimento das informações necessárias a estas três categorias (***'White Pages', 'Yellow Pages' e 'Green Pages'***).
A especificação UDDI inclui um ***'XML Schema'*** que define quatro tipos de informação principais, estando eles relacionados com a categorização da Figura 2-13.

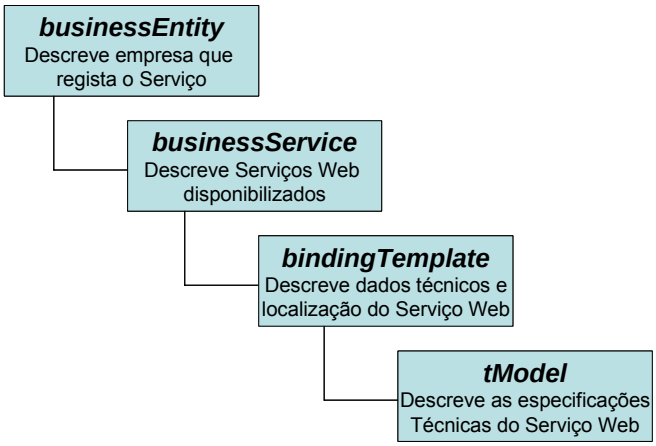


Figura 2-13 – XML Schema do UDDI

Na entidade de negócios ***'businessEntity'*** é possível descrever a informação relativa ao fornecedor do serviço, correspondendo à informação contida nas Páginas Brancas (***'White Pages')*** anteriormente referidas. A estrutura do ***'businessEntity'***, segundo a UDDI *Data Structure Reference V1.0*, encontra-se exemplificada na Figura 2-14.

```
<element name = "businessEntity">
    <type content = "elementOnly">
        <group order = "seq">
            <element ref = "discoveryURLs" minOccurs = "0" maxOccurs = "1" />
            <element ref = "name" />
            <element ref = "description" minOccurs = "0" maxOccurs = "*" />
            <element ref = "contacts" minOccurs = "0" maxOccurs = "1" />
            <element ref = "businessServices" minOccurs = "0" maxOccurs = "1" />
            <element ref = "identifiedBag" minOccurs = "0" maxOccurs = "1" />
            <element ref = "categoryBag" minOccurs = "0" maxOccurs = "1" />
        </group>
          <attribute name = "businessKey" minOccurs = "1" type = "string" />
          <attribute name = "operator" type = "string" />
          <attribute name = "authorizedName" type = "string" />
      </type>
  </element>
```

Figura 2-14 – Estrutura '*businessEntity*' do UDDI

O campo ***'name'*** fornece o nome da entidade fornecedora do Serviço WEB, ou seja, a empresa que está a realizar o registo.

No campo ***'description'*** pode ser armazenada uma breve descrição do fornecedor Serviço WEB.

No campo ***'contacts'*** pode ser guardada a informação relativa aos contactos da entidade que disponibiliza o serviço.

No campo ***'businessServices'*** pode ser guardada uma lista de todos os businessServices disponibilizados pela entidade fornecedora do serviço.

No campo ***'identifiedBag'*** encontra-se uma lista de identificadores alternativos para o fornecedor do Serviço WEB.

No campo ***'categoryBag'*** encontra-se uma lista de pares nome/valor, que identificam o fornecedor do serviço de acordo com a categoria pela qual o Serviço WEB pode ser procurado.

Da estrutura acima descrita, destaca-se o ***'businessKey'*** como sendo a chave do registo UDDI. Esta chave, em formato hexadecimal, permite definir univocamente o fornecedor do serviço e é-lhe atribuída pelo UDDI recorrendo a um algoritmo denominado UUID (Universal Unique Identifier).

O campo ***'operator'*** identifica o nome do UDDI onde o registo foi efectuado. Se o registo tiver sido efectuado na MICROSOFT, este campo terá a informação http://uddi.microsoft.com.

No campo ***'authorizedName'*** encontra-se o nome de quem regista as informações relativas à entidade fornecedora do serviço.

Na estrutura ***'businessService'*** (Figura 2-15), encontram-se as informações sobre cada um dos serviços oferecidos pela empresa, correspondendo à informação encontrada nas Páginas Amarelas (***'Yellow Pages'***). Para cada ***'businessEntity'*** podem existir vários ***'businessService'***, sendo este relacionamento feito através do campo ***'businessKey'***. A estrutura do ***'businessService'***, segundo a UDDI *Data Structure Reference V1.0*, é a que se apresenta como exemplo na Figura 2-15.

```
<element name = "businessService">
    <type content = "elementOnly">
        <group order = "seq">
            <element ref = "name" />
            <element ref = "description" minOccurs = "0" maxOccurs = "*" />
            <element ref = "contacts" minOccurs = "0" maxOccurs = "1" />
            <element ref = "bindingTemplates" />
            <element ref = "categoryBag" minOccurs = "0" maxOccurs = "1" />
        </group>
        <attribute name = "serviceKey" minOccurs = "1" type = "string" />
        <attribute name = "businessKey" type = "string" />
    </type>
</element>
```

Figura 2-15 – Estrutura *'businessService'* do UDDI

O campo ***'name'*** contém o nome do serviço em causa.

O campo ***'description'*** contém uma descrição das funcionalidades do serviço.

O campo ***'bindingTemplates'*** contém uma lista dos diversos ***'bindingTemplates'*** disponíveis no serviço.

No campo ***'categoryBag'*** encontra-se uma lista de pares nome/valor, que identificam o serviço de acordo com a categoria pela qual pode ser procurado.

No campo ***'serviceKey'*** encontra-se um atributo que permite definir univocamente o Serviço WEB.

No campo ***'businessKey'*** consta a chave da ***'businessEntity'*** que contém o serviço.

Na estrutura ***"bindingTemplate"*** (Figura 2-16) encontram-se descritas as informações técnicas sobre o Serviço WEB, correspondendo à informação encontrada nas Páginas Verdes (***"Green Pages"***). A estrutura do ***"bindingTemplate"***, segundo a UDDI *Data Structure Reference V1.0*, é a que se apresenta como exemplo na Figura 2-16.

```
<element name = "bindingTemplate">
    <type content = "elementOnly">
        <group order = "seq">
            <element ref = "description" minOccurs = "0" maxOccurs = "*" />
                <group order = "choice">
                    <element ref = "accessPoint" minOccurs = "0" maxOccurs = "1" />
                    <element ref = "hostingRedirector" minOccurs = "0" maxOccurs = "1" />
                </group>
                <element ref = "tModelInstanceDetails" />
        </group>
        <attribute name = "bindingKey" minOccurs = "1" type = "string" />
        <attribute name = "serviceKey" type = "string" />
</type>
</element>
```

Figura 2-16 – Estrutura do *'bindingTemplate'* do UDDI

No campo ***'description'*** encontra-se uma descrição do ***'bindingTemplate'.***

No campo ***'accessPoint'*** encontra-se a localização do Serviço WEB. Normalmente é um endereço URL, mas também pode ser um endereço e-mail ou número de telefone. O ***'accessPoint'*** possui um atributo ***'urlType'*** que indica o tipo de informação a que se refere (mailto, http, https, ftp, fax, phone, etc.).

No campo ***'hostingRedirector"*** só existe informação quando não é definido nenhum ***'accessPoint'***, passando assim a referenciar outro **'bindingTemplate'.**

No campo ***'tModelInstanceDetails'*** pode ser encontrada uma lista de elementos ***'tModelInstanceInfo'***, os quais contêm referências para elementos tModel. Essas referências são feitas através do atributo obrigatório tModelKey, o qual identifica univocamente o tModel ao qual se pretende referir.

Finalmente, na estrutura ***'tModel'*** pode ser encontrada a localização da informação técnica do Serviço WEB em causa. Na realidade, esta é utilizada sempre que é necessário indicar uma descrição que se encontra num ficheiro externo. A estrutura do ***'tModel'***, segundo a UDDI *Data Structure Reference V1.0*, é a que se apresenta como exemplo na Figura 2-17.

```
<element name = "tModel">
    <type content = "elementOnly">
        <group order = "seq">
            <element ref = "name" />
            <element ref = "description" minOccurs = "0" maxOccurs = "*" />
            <element ref = "overviewDoc" minOccurs = "0" maxOccurs = "1" />
            <element ref = "identifierBag" minOccurs = "0" maxOccurs = "1" />
            <element ref = "categoryBag" minOccurs = "0" maxOccurs = "1" />
        </group>
        <attribute name = "tModelKey" minOccurs = "1" type = "string" />
        <attribute name = "operator" type = "string" />
        <attribute name = "autorizedName" type = "string" />
    </type>
</element>
```

Figura 2-17 – Estrutura do *'tModel'* do UDDI

O campo ***'overviewDoc'*** contém a referência para a localização do ficheiro externo onde se encontra a especificação do Serviço WEB. Para tal conta-se com o sub-elemento ***'overviewURL'*** que indica a localização do ficheiro propriamente dito (o URL ou o WSDL).
Assim, tendo em conta as ligações entres as várias estruturas de dados do UDDI, pode-se construir o seguinte esquema de relacionamentos (Figura 2-18).

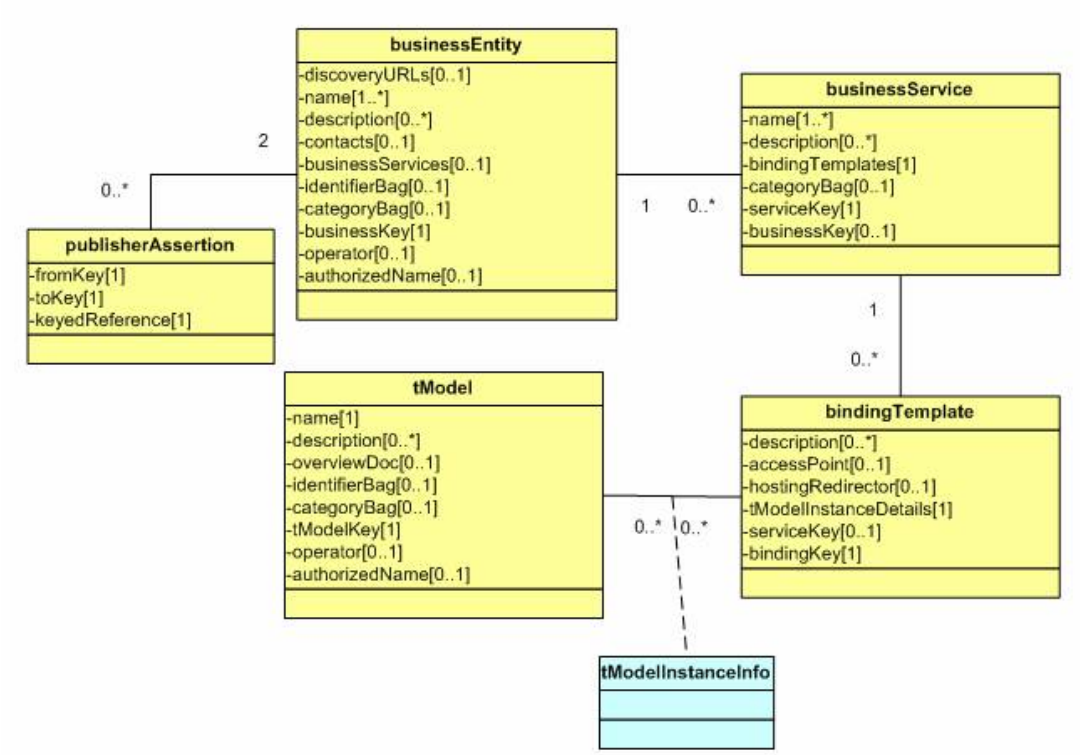


Figura 2-18 – Esquema de relacionamentos entre as estruturas do UDDI

2.1.4.3.3 Métodos do UDDI

Independentemente das plataformas ou linguagens de programação usadas, a fase de Publicação ou Divulgação consiste em criar, ler, actualizar ou apagar as informações relativas aos fornecedores e aos Serviços WEB existentes no directório UDDI. Assim podem ser chamadas as seguintes funções durante o processo de registo ou publicação (

Tabela 1). [DEV 2005]

Tabela 1 – Funções do UDDI na fase de Publicação

Método	Descrição
delete_binding	Remove um ou mais registos ***'bindingTemplate'*** do directório UDDI
delete_business	Remove um ou mais registos ***'business'*** do directório UDDI
delete_service	Remove um ou mais registos ***'businesService'*** do directório UDDI
delete_tModel	Remove um ou mais registos ***'tModel'*** do directório UDDI
save_binding	Grava ou actualiza o elemento ***'bindingTemplate'***
save_business	Grava ou actualiza informações respeitantes à estrutura ***'businessEntity'***
save_service	Grava ou actualiza o elemento ***'businessService'***
save_tModel	Grava ou actualiza o elemento ***'tModel'***

Durante a fase de pesquisa ou invocação são realizadas as operações de pesquisa de fornecedores ou dos serviços específicos. Na Tabela 2 podem ser observados alguns dos comandos utilizados para realizar estas operações.

Tabela 2 – Funções do UDDI na fase de Invocação

Método	Descrição
find_binding	Descobre 'bindings' existentes nos 'businessServices'
find_business	Localiza informações relativas aos ***'businesses'***
find_relatedBusinesses	Localiza informações relativas aos registos ***'businessEntity',*** registos estes que se encontram relacionados com um ***'business entity'*** específico cuja ***'key'*** foi obtida numa consulta
find_service	Descobre ***'services'*** específicos registados no ***'business entities'***
find_tModel	Descobre uma ou mais informações respeitantes aos ***'tModel'*** existentes na estrutura
get_bindingDetail	Recolhe informações do ***'bindingTemplate'***
get_businessDetail	Recolhe informações ***'businessEntity'***
get_businessDetailExt	Recolhe informações adicionais do ***'businessEntity'***
get_serviceDetail	Recolhe informações do ***'businessService'***
get_tModelDetail	Recolhe informações do ***'tModel'***

2.1.5 Considerações Sobre as Tecnologias de Suporte

Nesta secção vão ser feitas algumas considerações sobre as tecnologias de suporte atrás descritas, nomeadamente o DCE, CORBA e WEB Services (WSDL, UDDI e SOAP), como ferramentas/utilitários para dar respostas aos problemas inicialmente referidos no início nesta dissertação. Assim, pretende-se avaliar como estas tecnologias podem suportar a solução proposta no capítulo 1, durante a localização dos Servidores WEB (*Produtores*) e durante a identificação e a invocação dos seus serviços (Produtos).

O CORBA e o DCE foram concebidos para permitir que as aplicações distribuídas sejam independentes do hardware, dos sistemas operativos e das linguagens de programação.

As aplicações distribuídas que utilizam os WEB Services são também independentes do hardware e dos sistemas operativos. As aplicações distribuídas que utilizem os WEB services apenas necessitam de enviar e receber mensagens do tipo SOAP, utilizando para o efeito diversos protocolos de comunicação, nomeadamente os protocolos HTTP/TCP/IP.

Em SOAP/XML podem ser propostas "etiquetas" para dar resposta a novas necessidades, originadas por novas situações, ao passo que as plataformas distribuídas DCE e CORBA não são tão acessíveis. Um novo método ou função implica um registo deste no Directório de Serviços DCE ou IR (CORBA).

O CORBA é um componente orientado para objectos enquanto que os Serviços WEB e o DCE se preocupam mais com a mensagem em si, abstraindo-se da noção de objectos.

O DCE suporta ***'contexts'*** que são mecanismos que mantêm o estado do servidor durante o pedido de um cliente, isto é, o cliente pode estar a transmitir uma série de RPC para realizar uma operação ou acção individualmente. Estes ***'contexts'*** são geridos directamente e automaticamente pelo ***'Stub's'*** do cliente e do servidor. O CORBA não possui este mecanismo, sendo o programador responsável pela gestão da transmissão e recepção dos RPCs.

2.1.5.1 Localizar Servidores

Nesta fase pretende-se analisar o comportamento das tecnologias de suporte quando é necessário localizar um computador, existente numa rede distribuída, onde reside uma aplicação solicitada por um cliente. As aplicações cliente devem ser capazes de interagir com outras aplicações existentes numa rede distribuída de forma a executarem os serviços pretendidos, independentemente de conhecerem ou não as suas localizações. Devem também possuir mecanismos capazes de identificar e localizar esses serviços ou objectos.
Existe um componente no CORBA, chamado Directório de Implementação (Interface Repository), que contém as informações necessárias para permitir ao ORB localizar e processar um pedido de um determinado objecto a um servidor, mas apenas pelo nome que foi inicialmente identificado na rede. Nos Serviços WEB, o UDDI disponibiliza uma base de dados na WEB onde os serviços podem ser localizados pelo nome da entidade, pelo tipo de actividade ou pelo seu URL. Mostra-se assim uma forma mais completa de descobrir os serviços existentes e que podem ser invocados.

2.1.5.2 Identificar Serviços Fornecidos por Servidores

O CORBA possui um Directório de Implementação (Interface Repository) onde constam as informações respeitantes às operações que podem ser executadas num determinado objecto. A invocação desse objecto é feita usando o DII (Dynamic Invocation Interface).

No DCE é possível identificar os serviços no chamado directório de serviços DCE. A capacidade de localizar o serviço através deste directório é da responsabilidade do RPC DCE. Um cliente pode procurar um serviço pela sua proximidade, pelo seu UUID (Universally Unique Identifier), pelo nome ou até pelo seu tipo. No entanto, este repositório de informação pode não contemplar todos os serviços existentes na rede. Nos Serviços WEB é utilizada a especificação WSDL para definir na forma de um documento os serviços disponíveis, as variáveis utilizadas, as formas de interacção com o servidor que possui esse serviço, o protocolo de comunicação, etc. Este serviço mostra-se mais flexível no que diz respeito à introdução de novas funções ou tipos de dados a utilizar.

2.1.5.3 Invocar Serviços num Servidor

As aplicações que usam o RPC em sistemas computacionais distribuídos anteriores ao aparecimento da Internet, casos do DCE e do CORBA, podem encontrar problemas de compatibilidade e segurança durante a transmissão de dados quando se tentam adaptar aos novos meios de transmissão. Foi referido anteriormente que estas especificações possuem componentes que formatam automaticamente a mensagem a transmitir para a rede (***'Stub's'*** no cliente e no servidor). Estes problemas centram-se essencialmente no bloqueio da transmissão de dados por parte de ***'firewalls'*** e ***'proxys'*** existentes nos servidores. O uso da Internet, mais propriamente através do protocolo HTTP, é um excelente e o mais utilizado meio de transmissão de mensagens entre aplicações nos dias de hoje, visto o HTTP ser suportado pela maior parte dos browsers e servidores.

Os WEB Services já utilizam a Internet (HTTP) para solicitarem serviços aos servidores e receberem respostas destes, através de mensagens SOAP. Estas mensagens podem ser à primeira vista trabalhosas e por vezes complexas a criar, mas no final conseguem mostrar de uma forma clara e simples os parâmetros que se querem passar para a rede. Além disso, podem ser facilmente apresentadas nos Browsers WEB tendo que sofrer para isso uma simples transformação para HTML (HyperText Markup Language). Além do HTTP, as mensagens também podem ser transmitidas para a rede utilizando protocolos de comunicação como SMTP, e-mail, entre outros.

Tabela 3 – Comparação entre as tecnologias de suporte

	CORBA	DCE	WEB Services
Sistema	IDL (estático + verificação em tempo real)	IDL (dinâmico + verificação em tempo real)	XML Schemas (verificação em tempo real)
Sintaxe Transferência	CDR (formato binário)	CDR (formato binário)	XML (UTF-Unicode)
Estado	Statefull	Stateless	Stateless
Registo	Interface Repository Implementation Repository	Directório de serviços DCE	UDDI/WSDL
Serviço de localização	CORBA naming/trading service	Através UUID's no directório de serviços DCE	UDDI
Segurança	Plataforma CORBA	Plataforma DCE (Kerberos)	Assinaturas HTTP's, XML
Firewall	Não utilizado em HTTP	Não utilizado em HTTP	Através HTTP
Criação Mensagens	CORBA Stub's	RPCs Stub's	SOAP
Gestão erros	Excepções IDL	ACF (Attribute Configuration File) CATCHANY,CATCHALL	Mensagens Erro SOAP
Modelo Data	Objectos	Procedimentos	Mensagens SOAP
Ligação Cliente/Servidor	Directa	Directa	Indirecta
Parâmetros passados	Por referência e por valor	Pelo UUID	Por valor (não é tido em conta o objecto)
Meio Transmissão	GIOP/IIOP/TCP/IP	GIOP/IIOP/TCP/IP	HTTP/SMTP/TCP/IP
Linguagem Programação	Todas as que usem IDL	Todas as que usem IDL	Todas as que interpretem SOAP ou XML

2.2 Trabalhos Realizados Nesta Área de Conhecimento

Vários trabalhos nesta área e em áreas afins têm sido desenvolvidos por diversos autores. Nesta secção são apresentadas várias teses sobre esta temática, tal como os seus objectivos e as conclusões obtidas, de uma forma resumida.

Uma tese de mestrado intitulada "Web Service Interface and Architecture for Accessing FieldBus System" propõe um conjunto de serviços WEB para que seja possível aceder a uma rede de campo a partir da WEB, mais exactamente, ler e escrever valores em diversos dispositivos electrónicos ligados a essas redes de campo. O autor recorre também às normas WSDL para definir os serviços WEB que propõe e à norma SOAP para definir as operações e o formato das mensagens trocadas. Como conclusão, o autor refere que a solução por si proposta permite um elevado nível de abstracção dos dispositivos da rede de campo, da estrutura de dados e do seu formato. Refere ainda que uma arquitectura/interface WEB baseada em HTTP, SOAP pode ser facilmente acessível por outros sistemas e o código desenvolvido pode ser reutilizado com facilidade em diferentes plataformas computacionais. [Merino 2004]

Outra tese de mestrado intitulada "WebGraf – Aplicação Web para Execução de Grafcets e redes de Petri em Controladores Lógicos Programáveis" propõe uma ferramenta de tradução automática de GRAFCET ou redes de Petri para Lista de Instruções. Esta ferramenta está disponível na WEB e pode ser acedida a partir de um vulgar Browser WEB. Os programas escritos em GRAFCET ou Petri são previamente convertidos para um documento XML, usando um conjunto de "etiquetas XML", por ele propostas. Esse documento é então enviado para o servidor WEB onde reside a ferramenta de tradução proposta. A partir deste documento a ferramenta de tradução gera um programa autómato em lista de instruções. O algoritmo de conversão foi escrito em JSP (Java Server Pages). [Gomes 2003]

Outra tese de mestrado intitulada "Identificação e Controlo de Processos via Internet" propõe uma arquitectura que permite "a utilização remota de laboratórios pelos alunos, mesmo a partir de casa, sem preocupações de horário e sem preocupações com o eventual excesso de alunos que possam estar presentes no laboratório". Remotamente os alunos podem testar diversos algoritmos de controlo e observar o comportamento físico de vários processos no laboratório. Para permitir que um determinado sistema físico possa ser monitorizado e controlado remotamente via Internet, a arquitectura proposta utiliza um servidor WEB, um servidor de correio electrónico SMTP e um servidor de FTP, além de tecnologias COM, DDE, DAO, ADO para aceder aos recursos laboratoriais e às bases de dados para registar os resultados das

experiências. Foi possível ao autor concluir que esta solução permite que o controlo de sistemas pode ser ensinado recorrendo à utilização remota de laboratórios. [Silva 2003]

O projecto SIRENA, publicado na revista IEEE através de um artigo intitulado "Service-Oriented Paradigms in Industrial Automation" pretende abordar e lançar um desafio no desenvolvimento de dispositivos, aplicações e serviços, baseados nas novas tecnologias, sempre com o objectivo de aumentar e optimizar a inteligência e funcionalidades de que são providos. Faz uma descrição dos desafios enfrentados pela comunidade do ***"manufacturing"*** e das oportunidades resultantes da crescente tendência a pormenorizar e utilizar protocolos de comunicação padrão. É dada uma visão geral de como a representação dos padrões de modelos orientados a serviços, em particular quando são implementados utilizado WEB Services, podem ajudar a atingir os objectivos inicialmente propostos pelo autor. É também descrita e avaliada a estrutura de comunicação de um modelo orientado a serviços proposta pelo SIRENA (Service Infrastructure for Real-Time Embedded Network Applications) que permite o uso de dispositivos industriais em redes distribuídas. A evolução e a convergência entre a computação e as tecnologias das redes distribuídas, sustentadas pelos avanços das tecnologias dos semicondutores e da transmissão de dados, limitam a revolução de como as comunicações são organizadas entre sistemas e dispositivos. Certamente, o aparecimento e criação de dispositivos cada vez mais poderosos, fáceis de interligar e conectar entre si e a redes distribuídas permitirá integrar e utilizar a inteligência da computação e das comunicações numa programação de baixo nível (ao nível do dispositivo), possibilitando, apoiado pelo protocolo Internet, uma comunicação de alto nível. Seguindo esta tendência e com a necessidade de se adaptar ao rápido desenvolvimento de arquitecturas orientadas a serviços, baseadas em padrões dos WEB Services, o projecto SIRENA é implementado numa linguagem de alto nível que permite uma troca de dados e comunicação entre dispositivos, assim como entre dispositivos e aplicações. A abordagem feita pelo autor permite introduzir novas arquitecturas de dispositivos e permitirá de uma forma perfeita integrar arquitecturas de redes ao nível dos dispositivos, assim como ao nível de organizações. Para o autor, fica assim a promessa de prolongar o paradigma de sistemas de ***"manufacturing holonic"*** através da globalidade e diversidade de redes industriais. [James&Smit 2005]

Capítulo 3

Solução Proposta

3 SOLUÇÃO PROPOSTA

3.1 Introdução

A solução proposta, tal como foi referida no capítulo 1.4, propõe a utilização de um *Broker* para descobrir os *Produtores* capazes de fornecer a um *Cliente,* um produto ou serviço ao melhor preço e/ou o mais rapidamente possível, de forma simples, autónoma, rápida e automática. Este *Broker* deverá também conseguir negociar e encomendar o serviço ou produto pretendido segundo critérios previamente definidos pelo *Cliente*. Terá ainda que permitir, em qualquer instante, informar o *Cliente* do estado de produção de cada uma das suas encomendas. A solução proposta encontra-se ilustrada na Figura 3-1.

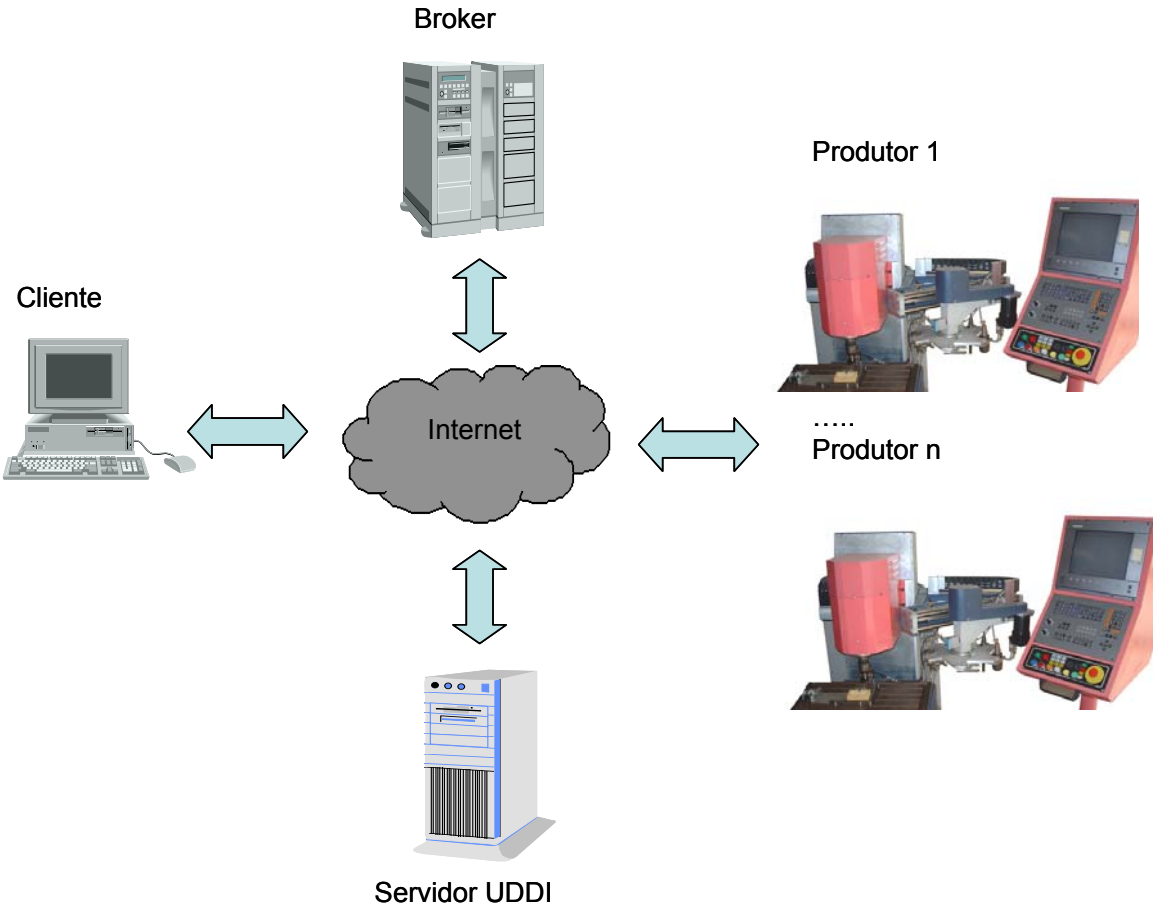


Figura 3-1 – Solução proposta

Um *Cliente* ao aceder ao endereço URL do *Broker* fica a saber quais os produtos que este pode e consegue negociar. Nessa altura, pode solicitar o orçamento para o fornecimento do produto que pretende, consoante critérios de prioridade (custo/prazo). O *Broker* tem que descobrir, recorrendo a servidores UDDI, quais os *Produtores* capazes de fornecer o produto em questão. Concluída a fase de pesquisa e após ter obtido uma lista de todos os *Produtores* capazes de fornecer esse produto, contacta cada um deles e, de uma forma automática, pede um orçamento para o produto solicitado pelo *Cliente*. O *Broker* fica à espera da resposta dos orçamentos dos vários *Produtores* e sempre que receber uma resposta contendo um orçamento, analisa-a e compara-a com as restantes. No final, deve apresentar ao *Cliente,* uma listagem com todos os orçamentos, incluindo o mais vantajoso consoante os critérios inicialmente definidos. O *Cliente* pode então escolher o orçamento que considera mais vantajoso e formalizá-lo através do seu pagamento pelos métodos disponibilizados pelo *Broker* (visa, cheque, transferência bancária, numerário, etc.). Ao receber a informação de que o pagamento da encomenda já foi efectuado (aspecto que não é tratado nesta dissertação e por isso assume-se que o pagamento foi sempre correctamente efectuado), o *Broker* contacta o *Produtor* que forneceu o orçamento, seleccionado pelo *Cliente,* e encomenda-lhe esse produto, escrevendo na sua base de dados de produção todas as informações necessárias. O *Produtor* ao finalizar a produção de uma encomenda envia-a ao *Cliente,* por correio ou DHL, e informa o *Broker* desse envio e conclusão da mesma. O pagamento ao *Produtor* é depois feito pelo *Broker* através de um método de pagamento por eles acordado. Desde o início da produção de uma encomenda até à sua conclusão, o *Cliente* pode fazer o seguimento do estado de produção dessa encomenda, tanto nas bases de dados do *Broker* como nas bases de dados dos *Produtores*. Assim, o *Cliente* contacta o *Broker* e este disponibiliza-lhe, de acordo com as informações existentes na sua base de dados, o estado de todas as encomendas. O Cliente pode ainda solicitar ao *Broker* o estado de uma encomenda em particular, encarregando-se este de entrar em contacto com o *Produtor* em causa, solicitando-lhe o estado de produção dessa encomenda.

3.2 Tecnologias de Suporte Utilizadas

No capítulo 2 foram apresentadas diversas tecnologias “middleware” que poderiam ser utilizadas para implementar as plataformas do *Broker* e do *Produtor*. Depois de devidamente ponderadas as vantagens e desvantagens das tecnologias de suporte, optou-se pela utilização dos WEB Services.

O uso de ***‘middlewares’*** baseados em protocolos de comunicação tem tido grande aceitação em redes locais e partilhadas, onde as restrições de segurança são adoptadas de acordo com

os ***'middlewares'*** utilizados nessas redes. No entanto, os **'*middlewares*'** adoptados nas redes distribuídas locais e partilhadas podem não conseguir ultrapassar os sistemas de segurança (**'*firewalls*'**) adoptados em redes que partilhem dados na Internet. As arquitecturas CORBA e DCE não conseguiram uma implantação massiva na Internet visto os protocolos utilizados não seguirem um padrão mundial, o que poderia causar problemas de interoperabilidade nas diversas implementações dos ORBs. Como um dos principais objectivos deste trabalho é descobrir e contactar os vários *Produtores* mundiais, capazes de fornecer um determinado produto utilizando a Internet como infra-estrutura de comunicação, adoptaram-se os WEB Services como tecnologia de suporte para a implementação da solução proposta. Assim, o meio de comunicação no qual se iriam transmitir os dados ou informações – a Internet e os seus sistemas de segurança – foi um parâmetro importante na escolha da tecnologia de suporte.

Um WEB Service tem como objectivo fornecer de um determinado serviço a outras aplicações distribuídas ou a outros WEB Services. Este serviço pode ser traduzido numa simples validação ou numa informação mais detalhada como pode ser encontrada na solução proposta (número do orçamento, prazo, custo, identificação do *Produtor)*. Para que a tecnologia de suporte adoptada seja viável e flexível deve prever mecanismos padrão de definição de transporte, localização e publicação de serviços. Estes mecanismos devem encontrar-se presentes nas plataformas onde se desejem integrar, serem fáceis e rápidos de implementar, transparentes e isentos de quaisquer custos e pagamentos de patentes. É aqui que reside a maior vantagem da utilização dos WEB Services, pois utilizam protocolos padrão da Internet (SOAP, HTTP, WSDL), um formato padrão de troca de informações (XML) e um directório de registo e localização (UDDI). São estes os factores que fornecem pré-requisitos importantes para a larga aceitação dos WEB Services face a outras arquitecturas, tais como o CORBA e o DCE, que se encontram ligadas a protocolos específicos e que precisam de ser conhecidos pelas camadas que pretendem comunicar entre si.

Uma outra grande vantagem na utilização destes WEB Services é a facilidade e rapidez de integração e implementação de novas funções ou serviços. Os WEB Services apresentam também uma grande melhoria no que diz respeito à segurança. Para isso, basta aceder e editar o documento WSDL, ficando os novos serviços disponíveis para as restantes aplicações, sem necessidade de registar novos objectos ou serviços. Verifica-se ainda uma maior segurança visto que ao *Cliente* só é fornecida a informação que ele solicitou, evitando-se a transmissão de tabelas de dados completas com informações ou dados desnecessários.

Apesar dos WEB Services serem mais utilizados quando se pretende disponibilizar ou trocar dados na Internet, uma vez que permitem uma comunicação assíncrona eliminando os problemas relacionados com *'**firewalls**'*, não apresentam um desempenho idêntico ao do CORBA e ao do DCE, como tecnologia de suporte, em sistemas distribuídos de redes locais

e/ou partilhadas. O tratamento dos documentos XML é bastante mais complexo e complicado quando comparado com os objectos distribuídos utilizados no CORBA ou DCE, que se mostram mais adequados na construção e implementação de sistemas de alto desempenho em redes locais e/ou partilhadas.

Uma desvantagem do uso de WEB Services é actualmente o facto de ser uma tecnologia recente e se encontrar pouco difundida, encontrando-se ainda em fase de desenvolvimento e padronização por parte da W3C.

A

Figura 3-2 descreve as mensagens trocadas entre as várias entidades envolvidas (*Cliente*, *Broker*, *Produtor*, *Servidor UDDI*) de forma a permitir o ***'e-Business'*****,** tornando mais simples, rápido e eficaz o pedido e fornecimento de orçamentos, realização e seguimento de encomendas.

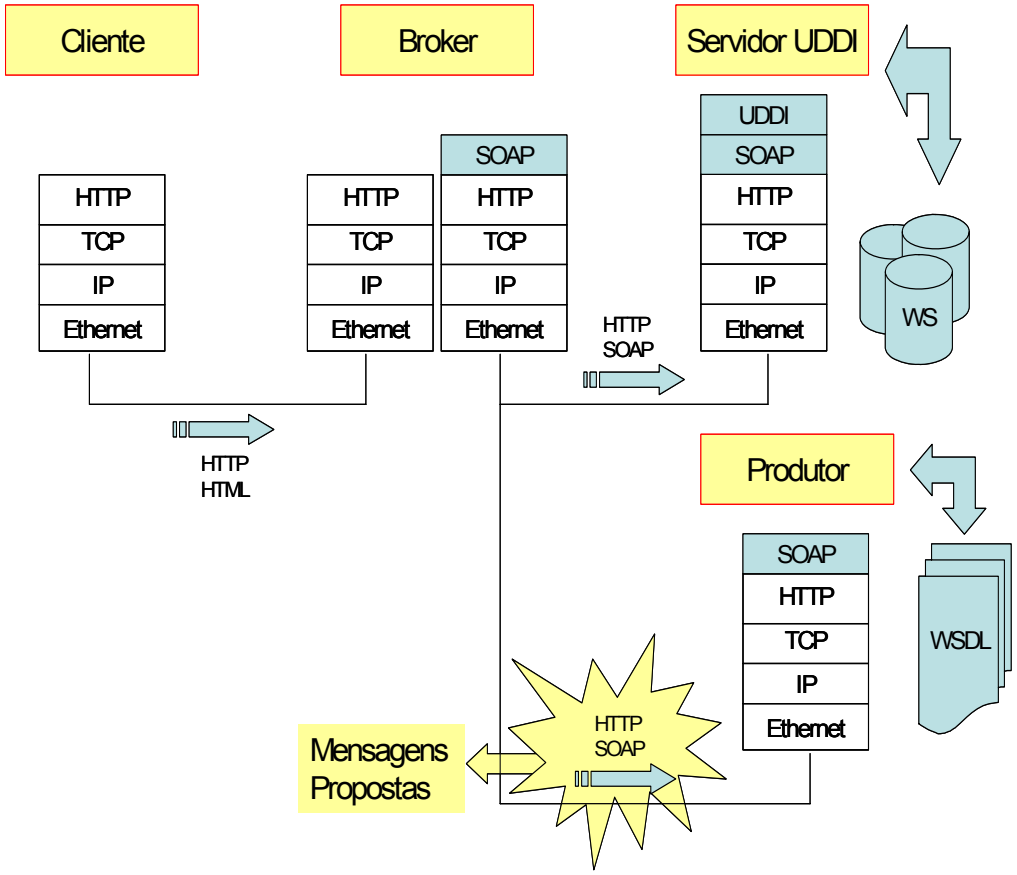


Figura 3-2 – Tecnologias de suporte utilizadas

O *Cliente* comunica essencialmente com o *Broker* através de mensagens HTML bidireccionais através do protocolo HTTP. O *Broker* vai comunicar com o *Servidor UDDI* e com os *Produtores* através de mensagens bidireccionais SOAP, igualmente através do protocolo HTTP. Será

através das mensagens SOAP entre o *Broker* e o *Servidor UDDI*, que o *Broker* descobrirá os *Produtores* que disponibilizam, a nível mundial, um determinado serviço. Os Produtores terão que registar, no *Servidor UDDI*, as informações pessoais e os serviços disponibilizados, recorrendo para isso a mensagens SOAP e utilizado o protocolo HTTP. Os *Produtores*, para além deste registo, ainda têm que disponibilizar um documento onde constam as informações relacionadas com os serviços disponibilizados, formas e parâmetros de interagir com as entidades externas. Este documento é redigido e apresentado recorrendo à especificação WSDL. Nos parágrafos seguintes será justificada e pormenorizada a escolha e utilização dos WEB Services.

Das várias mensagens presentes na
Figura 3-2 realçam-se as mensagens trocadas entre o Broker e os Produtores. Estas mensagens são o tema central desta tese, baseiam-se na especificação SOAP e são elas que permitem os serviços WEB propostos. Para que estas mensagens possam ser transferidas entre o Broker e os Produtores vários outros protocolos de comunicação são utilizados:

Protocolo de transferência – Dos vários protocolos de transferência de dados previstos pelos WEB services, escolhemos o protocolo HTTP para suportar o transporte das mensagens entre as várias entidades/aplicações desta arquitectura.

Estrutura das mensagens – o SOAP é a especificação adoptada pelos WEB Services responsável pela estrutura das mensagens a transmitir de forma a poderem ser entendidas e interpretadas por qualquer umas das aplicações distribuídas.

Documentos de descrição de serviços – O WSDL é a especificação adoptada pelos WEB Services para descrever os serviços WEB disponíveis na WEB. Neste caso os documentos WSDL descrevem os serviços propostos, disponibilizados pelos produtores.

Serviço de descoberta de serviços – O UDDI é a especificação adoptada pelos WEB Services para registar serviços WEB disponíveis em todo o mundo. No caso desta arquitectura, os servidores UDDI implementando esta especificação permitem aos *Produtores* registarem os seus serviços e ao *Broker* descobri-los.

3.3 *Broker* Proposto

O *Broker* proposto é uma aplicação informática (motor de busca avançado) capaz de descobrir, de uma forma automática, simples, rápida e autónoma, as entidades fabris ou fornecedores de serviços (*Produtores*) capazes de fornecer um determinado *Produto*.

Para uma maior flexibilização e acesso à informação existente, os serviços fornecidos pelo *Broker* estão disponíveis em páginas WEB, estáticas ou dinâmicas, podendo ser acedidas através de qualquer *Browser WEB.* Isto permite a utilização dos serviços do *Broker* em qualquer instante e em qualquer parte do mundo, bastando para isso recorrer a um simples computador, PDA ou telemóvel que possua um *Browser WEB* e uma ligação à WEB (Wireless LAN (Wireless Local Area Network) ou WAP (Wireless Application Protocol)).

Tem também a capacidade de pedir, negociar e avaliar junto dos vários *Produtores*, o melhor orçamento para a produção de um *Produto* que lhe seja solicitado por parte de um *Cliente*.
Ao receber o pedido de produção de um produto (orçamento escolhido pelo *Cliente*), tem a capacidade de contactar o *Produtor* em causa e escrever na sua base de dados de produção as informações necessárias à produção desse produto de uma forma autónoma e automática. Desde o início da fase de produção até à conclusão e entrega do *Produto* final nas instalações do *Cliente*, o *Broker* comporta-se ainda como um interlocutor, informando o *Cliente* do estado da sua encomenda em cada momento.

Através do Browser WEB, quando um *Cliente* solicita ao *Broker* um serviço a ser realizado num *Produtor,* é necessário que esse serviço seja conhecido por todas as entidades envolvidas. O *Broker,* por exemplo*,* não pode solicitar um serviço que não seja conhecido num Produtor. Para que haja uma correcta interoperacionalidade entre todas as entidades, tem que haver um correcto conhecimento de todos os serviços disponibilizados pela totalidade das entidades envolvidas. Para isso, o *Broker* disponibiliza quatro serviços principais: Localização de Fornecedores, Orçamentação e Negociação, Realização de Encomenda e Seguimento de Encomenda, que se aprofundarão no capítulo 3.4.3 (Serviços WEB Propostos).

3.3.1 Desenvolvimento

O *Broker* é uma aplicação informática desenvolvida em PHP (Personal Home Page) [PHP 2006], possui uma base de dados realizada em Microsoft ACCESS [ACCESS 2006] acedida e actualizada através de uma ligação ODBC (Open DataBase Connectivity) e um conjunto de páginas HTML que permitem uma melhor interacção com os *Clientes* finais, que acedam aos seus serviços através do endereço URL. O *Broker* pode ser encontrado no endereço http://ims.mec.ua.pt/broker. Foi utilizado o APACHE [APACHE 2005] como Servidor WEB. O Apache e o PHP foram os softwares escolhidos para a implementação desta plataforma WEB porque, após várias experiências onde foram realizadas várias medições, mostraram ter um desempenho mais rápido e fiável na realização de operações de processamento de dados, assim como na leitura e escrita em bases de dados Microsoft ACCESS. [Alvarinhas 2005]

Nestas experiências foram igualmente medidos os desempenhos do servidor WEB IIS (Internet Information Services) em conjunto com páginas ASP (Active Server Pages) e PHP. Para o desenvolvimento e criação dos Serviços WEB propostos e usados pelo *Broker,* foi usada uma biblioteca livre e gratuita desenvolvida em PHP, chamada NuSOAP. [NUSOAP 2006]

3.3.2 Base de Dados do Broker

Para armazenar toda a informação vinda do *Cliente*, do *servidor UDDI* e dos *Produtores* ao longo de todo o processo de localização, orçamentação, encomenda e seguimento de encomendas, o *Broker* necessita de uma base de dados devidamente estruturada. Esta base de dados encontra-se no servidor do *Broker* e foi desenvolvida em MsAccess. O seu acesso e escrita são feitos utilizando uma ligação ODBC, a qual foi previamente definida nas ligações ODBC com o nome "Servidores". A base de dados existente no *Broker* é composta por cinco tabelas que se relacionam da forma ilustrada na Figura 3-3.

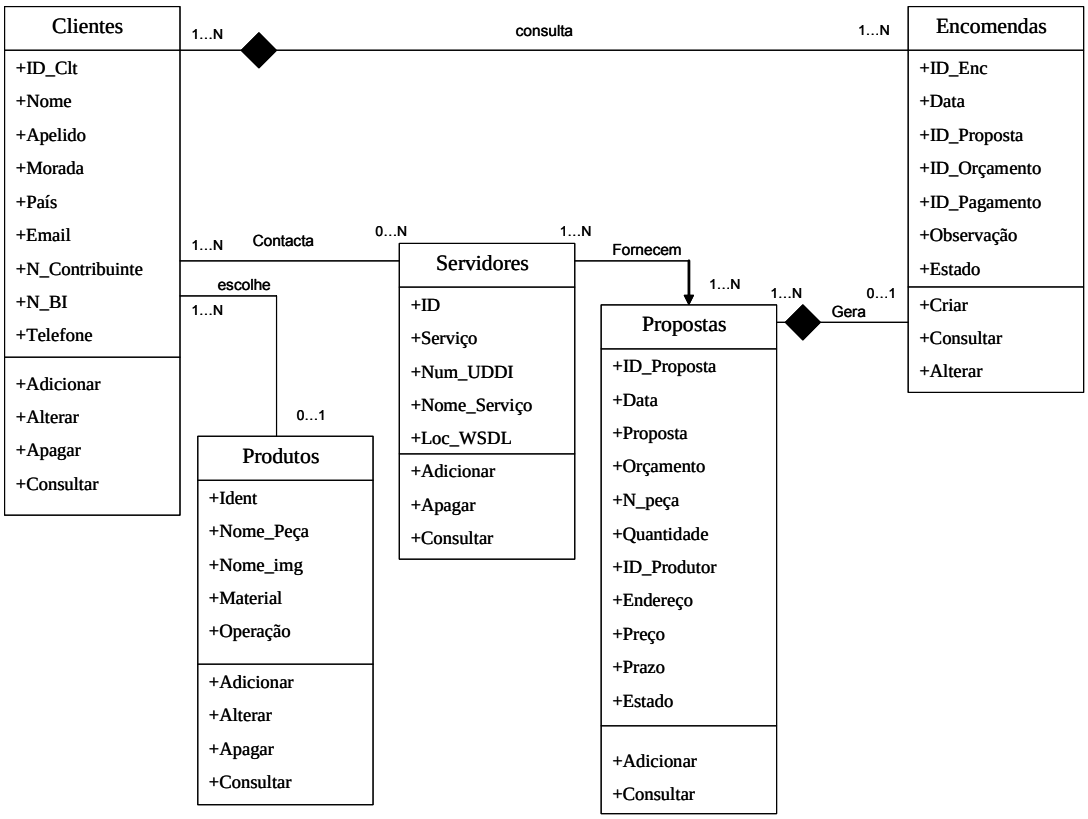


Figura 3-3 – Relacionamento das várias bases de dados

Onde nas tabelas:

Produtos – contêm as informações de todos os produtos disponibilizados pelo *Broker*.

Servidores – contêm as informações dos *Produtores* a contactar para cada um dos serviços requisitados pelo *Cliente*. Estas informações são provenientes da consulta do *Broker* ao servidor UDDI.

Propostas – contêm todas as propostas solicitadas pelos *Clientes* e os respectivos orçamentos devolvidos pelos *Produtores* contactados.

Clientes – contêm as informações relativas aos *Clientes* registados no *Broker*.

Encomendas – contêm as informações respeitantes às encomendas realizadas pelos C*lientes* e o estado das mesmas, em que o *Broker* é intermediário.

Na tabela "Produtos" encontram-se cinco campos que são preenchidos pelo gestor do *Broker*, e que correspondem aos produtos a serem disponibilizados pelo *Broker*:

Ident – é um campo do tipo texto que contém o número que identifica a peça a disponibilizar ao *Cliente*.

Nome_Peça – é um campo do tipo texto que contém o nome da peça. Este nome será o apresentado ao *Cliente* quando este aceder à lista de peças disponíveis pelo *Broker*.

Nome_Img – é um campo do tipo texto que contém o nome do ficheiro JPEG (Joint Photographic Experts Group) que, através de uma imagem ou fotografia, permite ao *Cliente* identificar a peça de uma forma mais fácil e intuitiva. Estes ficheiros encontram-se numa directoria chamada "peças" existente no servidor do *Broker*.

Material – é um campo do tipo texto que descreve o material de que é feita a peça em causa.

Operação – é um campo do tipo texto que identifica a operação a realizar para a obtenção da peça em causa.

A tabela "Servidores" é constituída igualmente por cinco campos e é preenchida automaticamente com o resultado da consulta aos servidores UDDI. O *Cliente* ao escolher uma peça está, de uma forma involuntária, a pedir ao *Broker* uma proposta com vários orçamentos.

O *Broker* terá de ser capaz de encontrar uma lista de *Produtores* capazes de realizar o pedido do *Cliente*. Assim, o *Broker* terá que consultar os vários servidores UDDI (IBM, MICROSOFT, GOOGLE) de forma a obter essa dita lista. O *Broker* à medida que vai obtendo as respostas dos servidores UDDI, vai preenchendo os vários campos dessa tabela. Estes campos serão depois consultados pelo *Broker* aquando a operação de pedido de propostas e orçamentos, realização e validação das encomendas e seguimento do estado das mesmas. Os campos que compõem esta tabela são os seguintes

ID – campo do tipo numeração automática que identifica o número do *Produtor* para uma determinada operação pedida pelo *Cliente*.

Serviço – campo do tipo texto que identifica a operação de selecção deste *Produtor*. Essa identificação pode ser feita pelo tipo de operação a realizar para uma determinada peça pedida pelo *Cliente*.

Num_UDDI – campo do tipo texto onde vai ser guardado o número que identifica o *Produtor* no Servidor UDDI (UDDI Number).

Nome_Serviço – campo do tipo texto que identifica o nome do *Produtor*, tal como se encontra registado no servidor UDDI.
Loc_WSDL – campo do tipo texto onde se encontra o endereço URL, pelo qual será posteriormente possível contactar um determinado *Produtor*.

Uma outra tabela proposta para a implementação do *Broker*, e que se pode ver no diagrama de relacionamentos das tabelas da base de dados, é a tabela das “Propostas”. Esta tabela é preenchida à medida que os vários *Produtores* vão respondendo com os seus orçamentos, após ter sido realizado um pedido de proposta de orçamento para a aquisição de um determinado produto por parte do *Cliente*. Assim, podem ser encontrados os seguintes campos:

ID_Proposta – campo do tipo numeração automática que identifica o número da proposta e o número de orçamento. Cada resposta de orçamento para uma determinada proposta é traduzida numa numeração que identifica o conjunto proposta/orçamento.

Data – campo do tipo texto onde fica guardada a data em que é gerada a proposta.
Proposta – campo do tipo texto que identifica o número da proposta. Cada solicitação de pedido de proposta/orçamento de uma peça por parte do *Cliente* é traduzida numa proposta que pode ter um ou mais orçamentos.

Orçamento – campo do tipo texto que guarda a resposta obtida de cada um dos *Produtores* contactados para a realização de uma determinada peça solicitada pelo *Cliente*. Se para uma peça, o *Broker* tiver solicitado o orçamento a vários *Produtores*, irão ser gerados vários orçamentos para uma só proposta.

N_peça – campo do tipo texto que guarda o número da peça solicitada pelo *Cliente.*

Quantidade – campo do tipo texto que guarda a quantidade de peças solicitadas pelo *Cliente* aquando do pedido de propostas e orçamentos.

ID_Produtor – campo do tipo texto onde fica guardado o número de identificação do *Produtor* que respondeu com um determinado orçamento.

Endereço – campo do tipo texto onde fica guardado o endereço URL ou a localização do ficheiro WSDL para onde o Broker solicitou o pedido de orçamento.

Prazo – campo do tipo texto onde se encontra o prazo estimado de cada *Produtor* para a realização e entrega da peça solicitada pelo *Cliente.*

Custo – campo do tipo texto onde se encontra o custo estimado de cada *Produtor* para a realização da peça solicitada pelo *Cliente*.

Antes de qualquer *Cliente* poder realizar uma encomenda de um determinado produto ou peça, tem que se registar na base de dados do *Broker*, tendo que fornecer um conjunto de informações que o identifiquem claramente ao realizar esse mesmo pedido. O *Cliente* ao registar-se no *Broker*, está a preencher a tabela "*Clientes*" que contém os seguintes campos:

ID_Clt – campo do tipo numeração automática que identifica o *Cliente* na base de dados do Broker.

Nome – campo do tipo texto que contém o nome próprio do *Cliente.*

Apelido – campo do tipo texto que contém o apelido ou o último nome do *Cliente.*

Morada – campo do tipo texto que contém a morada do *Cliente*. Esta morada será a morada para onde serão enviadas as encomendas acabadas.

País – campo do tipo texto que contém o nome do país do *Cliente.*

E-Mail – campo do tipo texto que contém o endereço electrónico (e-mail) do *Cliente*. O e-mail também vai servir para identificar o *Cliente* quando este pretender saber o estado das suas encomendas.

N_Contribuinte – campo do tipo texto que contém o número de contribuinte ou fiscal do *Cliente*, que será usado, por exemplo, para passar a factura ou o recibo quando efectuar os devidos pagamentos.

N_Bi – campo do tipo texto que contém o número do bilhete de identidade do *Cliente*.

Telefone – campo do tipo texto que contém o número de telefone ou telemóvel do *Cliente*.

O *Cliente* após ter recebido a proposta com os orçamentos dos vários *Produtores* e depois de estar correctamente registado no *Broker*, encontra-se finalmente em condições de realizar a encomenda da peça pretendida. Para isso tem que memorizar o nome da proposta e do orçamento que achou mais vantajoso. Ao preencher os vários campos na página WEB Encomendas do *Broker*, está a preencher a tabela "Encomendas" e os seguintes campos:

ID_Enc – campo do tipo texto que contém o número da encomenda na base de dados do *Broker*.

Data – campo do tipo texto que contém a data em que o *Broker* recebeu a encomenda na sua base de dados.

ID_Proposta – campo do tipo texto que contém o número da proposta relativa à peça a encomendar. Este número é o que corresponde ao conjunto proposta/orçamento escolhido.

ID_ Clt – campo do tipo texto que contém o número que identifica o *Cliente*. Este campo é preenchido através do e-mail que o *Cliente* tem que introduzir no início do formulário da encomenda.

ID_Pagamento – campo do tipo texto que contém a forma de pagamento escolhida pelo *Cliente* para pagar os serviços solicitados.

Observação – campo do tipo texto que pode conter ou não observações variadas.

Estado – campo do tipo texto que contém o estado da encomenda, isto é, se aguarda pagamento, se já foi enviada para o *Produtor* ou se já se encontra concluída.

3.4 Os Produtores

Nesta secção descreve-se a implementação e o tipo de *Produtores* utilizados. Como não foi possível usar uma instalação fabril completa para implementar e avaliar a solução proposta, foi seleccionado apenas um recurso fabril, mais concretamente um centro de maquinação. A este recurso iremos chamar *Produtor Real*.

Para possibilitar o processo de pedido e negociação, bem como a escolha do orçamento economicamente e/ou temporalmente mais vantajoso, foram criados *Produtores* Simulados. Estes têm como objectivo fornecer propostas de orçamentos aleatórias e concorrentes às propostas do *Produtor Real* (centro maquinação UA).

3.4.1 Tipos de Produtores

3.4.1.1 Produtores Reais

Nesta secção descreve-se a implementação dos *Produtores Reais* e a forma como os seus recursos podem ser disponibilizados na WEB. O recurso disponibilizado pelo *Produtor Real* trata-se de um centro de maquinação cuja função é produzir ou maquinar as várias peças disponibilizadas na página do *Broker*. Mais à frente irá ser apresentada a estrutura mecânica do centro de maquinação, os eixos, o armazém de ferramentas e o comando numérico (CN) que o controla, a interface de comunicação que o recurso disponibiliza para o exterior, nomeadamente as ligações eléctricas e os protocolos de comunicação. É através desta interface que um computador local consegue comunicar com o CN para o monitorizar e o controlar localmente. Neste capítulo descreve-se igualmente a forma como foi possível disponibilizar o centro de maquinação na WEB, de modo a poder receber e processar automaticamente pedidos remotos.

3.4.1.1.1 Centro de Maquinação

O centro de maquinação ou recurso do *Produtor Real* que se disponibilizou na WEB, encontra-se ilustrado na Figura 3-4 e é uma fresadora vertical com três eixos servo controlados e um armazém rotativo onde podem ser armazenadas até seis ferramentas, sendo a troca feita de

forma automática. O centro de maquinação é controlado por um CN da Heidenhain 426PB com disco rígido incorporado, simulação gráfica da maquinação.

Figura 3-4 – Recurso disponibilizado pelo *Produtor Real*

O CN possui dois tipos de interfaces de comunicação, a interface RS232 e a Ethernet /TCP/IP (Transmission Control Protocol/Internet Protocol). Através destas interfaces é possível trocar mensagens de texto nos formatos e sequências definidos pelo protocolo LSV2, transferir programas peça para o NC, dar ordem de maquinação e monitorizar diversos parâmetros de maquinação. Contudo, nem todas estas interfaces, nem o protocolo LSV2 permitem a localização e negociação remotas necessárias à sub-contratação. É necessário utilizar um computador local para fornecer os serviços WEB propostos nesta dissertação. Este computador conjuntamente com os programas desenvolvidos e as bases de dados definidas permitem implementar a plataforma WEB do *Produtor*.

3.4.1.1.2 Interface Rs232

A comunicação entre o CN e o computador (Servidor *Produtor*) pode ser realizada através da interface RS232 (Figura 3-5). Esta interface tem uma taxa de transferência de 110 bps a 115200 bps, 8 bits, sem paridade, 1 Stop Bit e o controlo de fluxo é feito por Hardware.

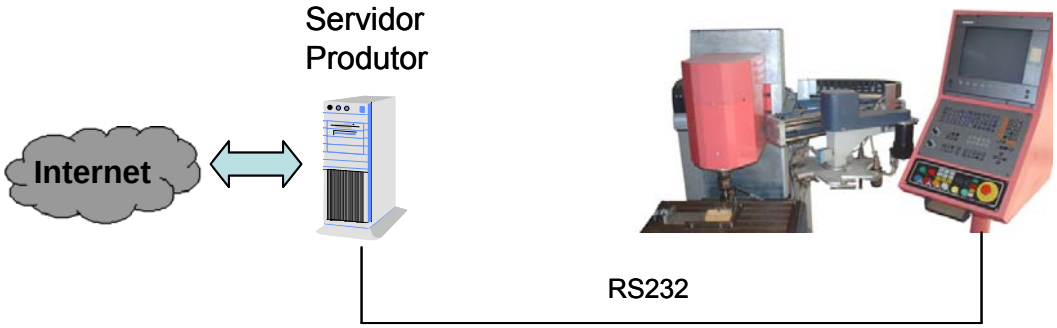


Figura 3-5 – Disponibilização do CN na WEB

3.4.1.1.3 Interface Ethernet

Esta comunicação é uma opção no CN e apenas se encontra disponível quando instalada uma placa de rede 10BaseT. O recurso pode ser acedido remotamente comportando-se como um cliente de uma rede local. O CN transmite as informações através da placa de rede recorrendo ao protocolo TCP/IP baseado no NFS (Network File System). Este tipo de interface é muito utilizado em sistemas UNIX onde os protocolos TCP/IP e NFS podem ser implementados sem recorrer à utilização de software adicional e específico. Nos computadores com sistemas operativos Windows o Protocolo TCP/IP é muito utilizado, contrariamente ao protocolo NFS. Para isso vai ser necessário um software adicional para ligar o CN ao servidor da rede local. Estas cartas de rede podem ser ligadas através de conectores RJ45 ou BNC, em ligação directa ou em LAN.

3.4.1.1.4 Protocolo LSV2

Através da ligação Rs232 ou Ethernet, descritas nas secções anteriores, pode ser utilizado o protocolo LSV2. Este permite a transferência de comandos, controlo e monitorização do CN a partir de um computador recorrendo ao envio de mensagens específicas. Estas mensagens, chamadas de telegramas, são blocos provenientes de um tratamento que consiste na separação dos dados a transmitir. A transferência destes telegramas pode ser dividida em quatro fases:

Fase de Repouso – o NC e o computador não estão a comunicar.

Fase de Inquérito – quando o computador pretende iniciar a comunicação com o NC, envia o byte <ENQ> (Enquiry – código 5 na tabela ASCII) e fica à espera de uma resposta do NC. Se o

computador não receber uma resposta válida após um determinado período de tempo, o protocolo LSV2 prevê o reenvio do byte <ENQ>.

Fase de Transferência de Dados – nesta fase, o NC já enviou a resposta ao computador e é iniciada a transferência do telegrama. O telegrama pode ter no máximo 128 bytes ASCII, onde os quatro primeiros bytes estão reservados ao código do telegrama. Este começa com o byte <STX> (Start of Text – código 02 na tabela ASCII) e termina com o byte <ETX> (End of Text – código 03 na tabela ASCII). O byte <DLE> (Data Link Escape – código 16 na tabela ASCII) é enviado no início dos caracteres <STX> e <ETX> e permite assegurar que o receptor reconheça o início e o fim da palavra.

Fase Final – nesta fase e após a transferência de dados ter sido concluída, o computador e o NC retornam à fase de repouso após ter sido encontrado o caracter <EOT> (End of Transmission – código 04 na tabela ASCII).

Na Figura 3-6 está representado um fluxograma onde são descritas as quatro fases envolvidas durante a transmissão de dados, utilizando o protocolo LSV2. [Dias 2006]

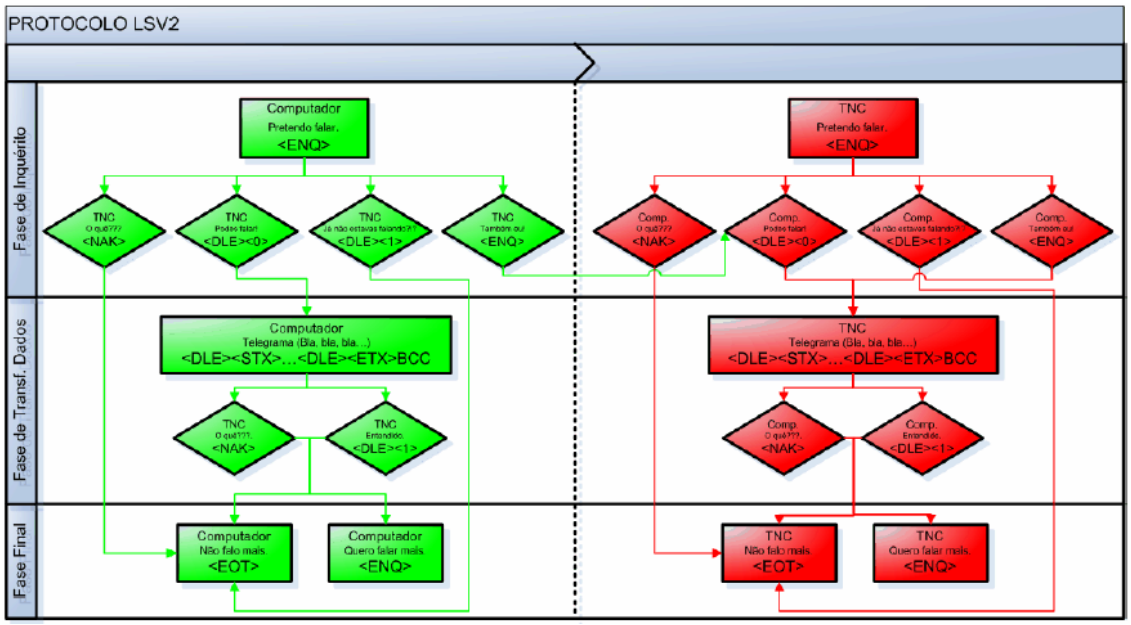


Figura 3-6 – Protocolo LSV2

Contudo, este e outros protocolos são por vezes complexos, de difícil compreensão e utilização. Para que programas de supervisão e controlo de recursos fabris possam ser facilmente implementados, os fabricantes disponibilizam softwares que permitem, a aplicações Windows, aceder aos recursos em causa. Estes softwares podem assumir a forma de:

1. **Bibliotecas de software (Dll, ActiveX)** que podem ser referidas ou incluídas no código dos programas de supervisão e controlo, durante o seu desenvolvimento.

2. **Aplicações Windows** fornecidas pelos fabricantes, que podem comunicar com os seus recursos. Estas aplicações não só comunicam com os seus recursos através das interfaces de comunicação do computador (porta série, Ethernet, etc.), como também podem comunicar com outras aplicações Windows através de ligações OPC e DDE, ligações essas internas ao ambiente Windows.

Desta forma, as aplicações Windows de supervisão e controlo apenas necessitam de aceder às aplicações dos fabricantes para, através delas, controlarem e monitorizarem os recursos fabris.

Neste caso foi utilizado um ActiveX fornecido pela Heidenhain, embora também pudesse ter sido utilizado um ActiveX desenvolvido no Departamento de Engenharia Mecânica da Universidade de Aveiro, no âmbito da disciplina de Informática Industrial. [Dias 2006]

3.4.1.2 Produtores Simulados

Nesta secção descreve-se a razão da existência e da forma como foram implementados os *Produtores Simulados*. A necessidade da criação e utilização de *Produtores Simulados* surgiu devido à inexistência de outros recursos físicos disponíveis (centros de maquinação ou *Produtores Reais*) para integrar o sistema de "Localização, Negociação e Contratação de Serviços através da WEB". Estes *Produtores Simulados* não são nada mais nada menos que aplicações desenvolvidas em VisualBasic, cujo objectivo é fornecerem propostas de orçamento quando são contactados pelo *Broker*. Cada um destes *Produtores Simulados* encontra-se registado no servidor UDDI podendo desta forma ser localizado e contactado pelo *Broker.* Assim, torna-se possível o processo de pedido e negociação de orçamentos entre os vários *Produtores* (Reais ou Simulados), onde no final é apresentado ao *Cliente* o orçamento mais vantajoso para o produto pretendido. Podendo estar disponíveis em qualquer parte do mundo, os *Produtores* são identificados na WEB pelo seu endereço DNS ou endereço TCP/IP.

Neste caso em particular, para os *Produtores Simulados,* foram atribuídas portas específicas geridas pelo servidor WEB APACHE (*Produtor1* atribuiu-se a porta82, *Produtor2* atribuiu-se a porta 83,..., e ao *Produtor5* atribuiu-se a porta 86). Estes *Produtores Simulados* possuem, da mesma forma que os *Produtores* Reais, uma base de dados de produção, um documento

WSDL onde estão definidos os serviços disponibilizados e um ficheiro PHP. Assim um *Produtor Simulado* ao receber na sua porta específica um pedido de orçamento, solicitado pelo *Broker*, vai devolver-lhe o seu documento WSDL informando-o dos serviços WEB disponibilizados. O *Broker*, antes de solicitar a produção de um *Produto* a um *Produtor*, guarda na sua base de dados "Encomendas" todas as informações respeitantes a essa encomenda.
As encomendas passam por vários estados desde que são escritas, pelo *Broker,* na base de dados de produção de cada um dos *Produtores Simulados*, até à conclusão de cada uma delas. Quando um *Produtor* recebe uma encomenda na sua base de dados de produção, atribui-lhe o estado de "Fila de Espera", informando de seguida o *Broker* da sua correcta recepção e tratamento. O *Broker* ao receber esta confirmação actualiza na sua base de dados "Encomendas" o estado dessa encomenda com "Enviado ao Produtor". O *Produtor Simulado* está constantemente a ler a sua base de dados "Produção", e quando não se encontra com uma encomenda em execução (simulação), mas com um registo no estado de 'Fila de Espera", simula a entrada em produção. Quando a encomenda entra em produção, o *Produtor* actualiza na sua base de dados o estado dessa encomenda para "Em Produção". O *Produtor* ao concluir a encomenda vai novamente actualizar na sua base de dados de produção o estado dessa encomenda para "Concluída", actualizando de igual forma o estado da mesma encomenda na base de dados "Encomendas" do *Broker*.

3.4.2 Bases de Dados do Produtor

A plataforma WEB de cada *Produtor* (*Real* e *Simulado*), para além das suas aplicações específicas, bases de dados privadas e outros softwares específicos, têm também no seu servidor uma base de dados, acedida através de uma ligação ODBC. Esta base de dados tem que estar definida no gestor de ligações ODBC e é constituída por uma tabela chamada Prod_Fornxxx, em que xxx será o nome ou identificação do *Produtor*. É nesta base de dados que o *Broker* escreve as encomendas por ele negociadas e que foram atribuídas a esse mesmo *Produtor*. O *Cliente* ao escolher e ao efectuar o pagamento da proposta/orçamento que achou mais favorável, informa o *Broker,* de uma forma indirecta, do *Produtor* que escolheu para a realização desse produto. O *Broker* vai contactar e escrever na sua base de dados do *Produtor* escolhido, as informações respeitantes à encomenda solicitada pelo *Cliente*. A tabela Prod_FornXXX (Figura 3-7) vai então ser preenchida nos seguintes campos:

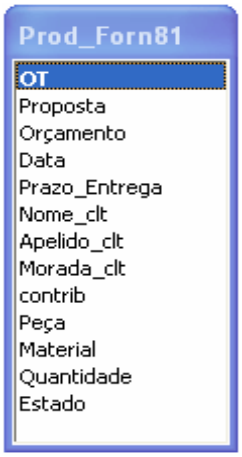


Figura 3-7 – Base de Dados do *Produtor*

Onde:

OT – campo do tipo numeração automática que vai identificar a ordem de trabalho ou o número do pedido inserido nessa tabela.

Proposta – campo do tipo texto que identifica o número da proposta escolhida pelo *Cliente* gerada pelo *Broker.*

Orçamento – campo do tipo texto que identifica o número do orçamento escolhido pelo *Cliente* gerado pelo *Broker.*

Data – campo do tipo texto que identifica a data e a hora a que essa encomenda ou ordem de trabalho foi inserida na base de dados do *Produtor.*
Nome_Clt – campo do tipo texto que guarda o nome do *Cliente* que efectuou a encomenda, para mais tarde endereçar a encomenda final por correio ou por transportadora.

Apelido_Clt – campo do tipo texto que guarda o apelido do *Cliente* que realizou a encomenda. Servirá mais tarde, tal como o nome, para identificar o destinatário da encomenda.

Morada_Clt – campo do tipo texto que guarda a morada do *Cliente* que realizou a encomenda, para endereçar a morada onde, posteriormente, irá ser entregue a encomenda.

Contrib – campo do tipo texto que guarda o número de contribuinte do *Cliente* que realizou a encomenda e que servirá mais tarde para preencher o recibo ou a factura relativa à encomenda.

Peça – campo do tipo texto que identifica a peça que o *Cliente* escolheu e que o *Produtor* deverá realizar.

Material – campo do tipo texto que indica o material que será usado para realizar a peça escolhida pelo *Cliente.*

Quantidade – campo do tipo texto que indica ao *Produtor* a quantidade de peças que deverá realizar e entregar ao *Cliente.*

Estado – campo do tipo texto que indica o estado de produção dessa encomenda ou ordem de trabalho, isto é, se a encomenda se encontra em *"Fila de Espera"*, se já entrou em *"Produção"* ou se já se encontra *"Concluída".*

3.4.3 Serviços WEB Propostos

Além da arquitectura proposta nesta dissertação prever a existência de um *Cliente*, de um *Broker* e de vários *Produtores*, são também propostos serviços WEB que permitem alcançar os objectivos anteriormente referidos. Estes serviços encontram-se definidos na linguagem WSDL. Tanto o *Broker* proposto como a plataforma WEB de cada *Produtor* têm de respeitar estas definições. O *Broker* actua como um *Cliente* destes serviços e a plataforma WEB como servidor, implementando-os. Mais exactamente, quando o *Cliente* pretende um determinado serviço do *Produtor,* é o *Broker* que vai gerar as mensagens SOAP adequadas de acordo com estes serviços WEB (WSDL). Cada mensagem SOAP identifica o nome do serviço pretendido e contém os parâmetros necessários a cada serviço. Quando a plataforma WEB executa o serviço pedido pelo *Broker*, o *Produtor* responde-lhe, enviando-lhe outra mensagem SOAP.

3.4.3.1 WSDL dos Produtores

Nos parágrafos seguintes é apresentado e descrito um documento WSDL existente no servidor de um dos *Produtores*. Como foi referido anteriormente, os documentos WSDL são escritos em XML, sendo por isso especificados como tal no início da sua declaração.

```
<?xml version="1.0" ?>
```

Um ficheiro WSDL pode ser caracterizado como sendo um documento de definições, logo existe um elemento chamado *definitions*, onde é definida a raiz do documento WSDL. O endereço URL (http://schemas.xmlsoap.org/soap/envelope) utilizado, não necessita de existir realmente, sendo sim obrigatório que seja único dentro do ficheiro WSDL. Dentro deste

elemento, podem estar contidos outros elementos (*types*, *message*, *portType*, *binding* e *service*). A etiqueta *targetNamespace* permite definir um *namespace* e o atributo *xmlns* permite referenciar outros *namespaces*.

```
<definitions
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:si="http://soapinterop.org/xsd"
xmlns:tns="urn:seguimentowsdl"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="urn:seguimentowsdl">
```

Um dos elementos que se encontra dentro do elemento *definitions* é o elemento *message*. Este elemento permite uma descrição abstracta das mensagens entre o *Cliente* e o *Produtor*. Cada mensagem contém uma ou mais *parts* que descrevem os parâmetros dos conteúdos das mensagens. Como se pode ver no código seguinte, são identificadas as mensagens através da etiqueta *message nam*” (*orcamentoRequest*, *orcamentoResponse*, *encomendaRequest*, *encomendaResponse*, *seguimentoRequest* e *seguimentoResponse*) e os parâmetros através da etiqueta *part name* necessários para as invocar sendo precedida da instrução *type* onde são definidos os tipo de dados.

```
<message name="orcamentoRequest">
  <part name="produto" type="xsd:string" />
  <part name="quantidade" type="xsd:string" />
  <part name="serv" type="xsd:string" />
  <part name="material" type="xsd:string" />
 </message>

<message name="orcamentoResponse">
  <part name="return" type="xsd:string" />
 </message>

…
```

```
…

<message name="encomendaRequest">
  <part name="name" type="xsd:string" />
  <part name="apelido" type="xsd:string" />
  <part name="morada" type="xsd:string" />
  <part name="contrib" type="xsd:string" />
  <part name="prop" type="xsd:string" />
  <part name="orcamento" type="xsd:string" />
  <part name="peca" type="xsd:string" />
  <part name="material" type="xsd:string" />
  <part name="quantidade" type="xsd:string" />
  <part name="prazo" type="xsd:string" />
 </message>

<message name="encomendaResponse">
  <part name="return" type="xsd:string" />
 </message>

<message name="seguimentoRequest">
  <part name="prop" type="xsd:string" />
  <part name="orcamento" type="xsd:string" />
  <part name="encomenda" type="xsd:string" />
 </message>

<message name="seguimentoResponse">
  <part name="return" type="xsd:string" />
 </message>
```

Um outro elemento que se encontra dentro do elemento *definitions* é o elemento *portType*, que define um grupo de operações para cada porta, isto é, é uma interface para o serviço. Cada operação é constituída por combinações de elementos de *input* e *output*. Assim, para cada mensagem trocada entre o *Cliente* e o *Produtor* é definida qual a mensagem (*Message Name*) que se utiliza para a mensagem de chamada e para a mensagem de resposta.

```
<portType name="seguimentowsdlPortType">
 <operation name="orcamento">
  <input message="tns:orcamentoRequest" />
  <output message="tns:orcamentoResponse" />
 </operation>

<operation name="encomenda">
 <input message="tns:encomendaRequest" />
 <output message="tns:encomendaResponse" />
</operation>

<operation name="seguimento">
 <input message="tns:seguimentoRequest" />
 <output message="tns:seguimentoResponse" />
</operation>
</portType>
```

Outro elemento que se encontra dentro do elemento *definitions* é o elemento *binding*. Este descreve os detalhes concretos de um determinado *portType* com um determinado protocolo, isto é, descreve o protocolo e o formato de dados para operações e mensagens definidas pelo p*ortType* (neste exemplo utiliza-se HTTP).

```
<binding name="seguimentowsdlBinding"
  type="tns:seguimentowsdlPortType">

<soap:binding style="rpc"
 transport="http://schemas.xmlsoap.org/soap/http" />
```

Tendo em conta a simples definição do elemento *message*, verifica-se que não é de todo possível associar mensagens de forma a obter-se um par pedido/resposta. O elemento *operation* é o responsável por indicar qual a mensagem correspondente ao pedido e a mensagem correspondente à resposta. Isto é conseguido através da etiqueta *input*, para as mensagens de pedido e *output,* para as mensagens de resposta.

```
<operation name="orcamento">
  <soap:operation soapAction="urn:orcamentowsdl#orcamento" style="rpc" />
     <input>
       <soap:body use="encoded" namespace="urn:orcamentowsdl"
         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
     </input>
     <output>
       <soap:body use="encoded" namespace="urn:orcamentowsdl"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
      </output>
  </operation>

<operation name="encomenda">
  <soap:operation soapAction="urn:encomendawsdl#encomenda" style="rpc" />
     <input>
       <soap:body use="encoded" namespace="urn:encomendawsdl"
         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </input>
    <output>
       <soap:body use="encoded" namespace="urn:encomendawsdl"
         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
     </output>
  </operation>

<operation name="seguimento">
  <soap:operation soapAction="urn:seguimentowsdl#seguimento" style="rpc" />
    <input>
       <soap:body use="encoded" namespace="urn:seguimentowsdl"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
     </input>
     <output>
       <soap:body use="encoded" namespace="urn:seguimentowsdl"
         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
       </output>
 </operation>
```

Um *port* é responsável por identificar o endereço do Serviço WEB ao qual se pretende aceder e que normalmente é atribuído o seu endereço URL. Cada *port* tem um atributo *binding* que não é mais do que uma referência a um elemento *binding* previamente definido. Dentro do

elemento *service*, encontra-se então o conjunto de *ports* que representam o Serviço WEB que se pretende disponibilizar.

```
<service name="seguimentowsdl">
 <port name="seguimentowsdlPort" binding="tns:seguimentowsdlBinding">
  <soap:address location="http://localhost/broker/somaserver81.php" />
 </port>
</service>
</definitions>
```

3.5 Descrição dos Fluxos de Informação

Nesta secção descrevem-se todos os fluxos de informação trocados entre o *Cliente*, o *Broker*, o *servidor UDDI* e os *Produtores*, desde que o *Cliente* pede ao *Broker* um simples orçamento para um determinado produto até o receber nas suas instalações. Esta solução prevê existência de quatro fases distintas, como pode ser visto na Figura 3-8: o *Registo de Produtores* (a amarelo), o *Pedido de Orçamento e Negociação* (a azul), a *Realização de Encomendas* (a vermelho) e o *Seguimento da Encomenda* (a verde).

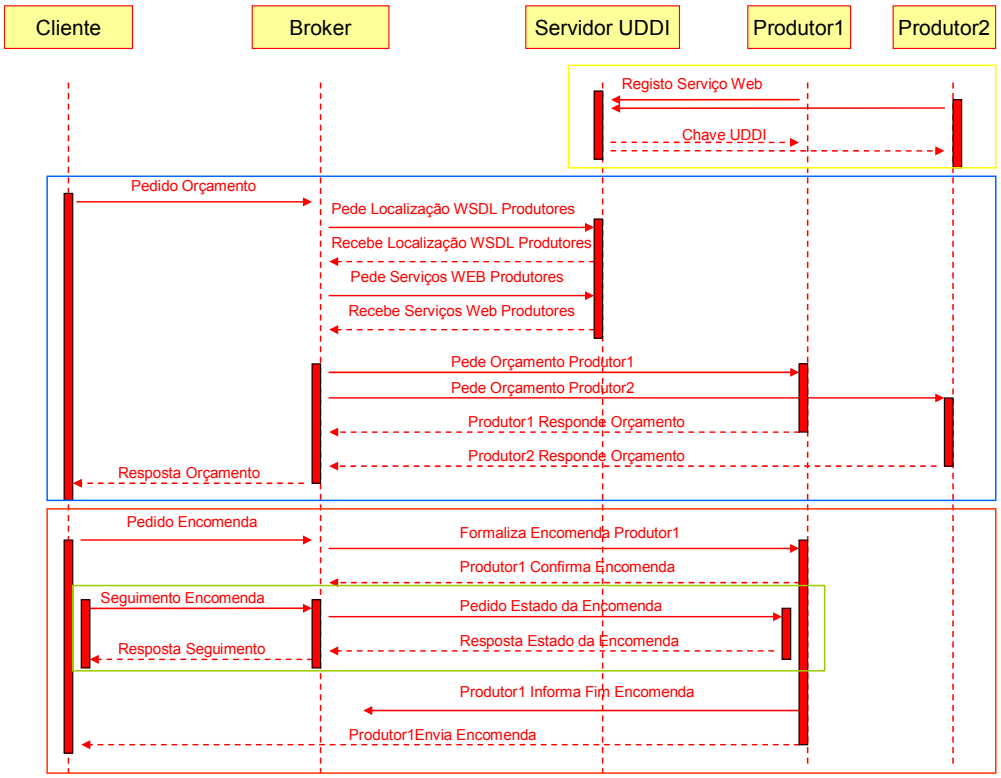


Figura 3-8 – Esquema temporal

De uma forma sucinta, quando um *Cliente* recorre aos serviços do *Broker* para lhe descobrir os *Produtores* capazes de prestar um serviço ao menor custo possível ou no prazo mais curto, o *Broker* consulta os *servidores UDDI* existentes na WEB e contacta cada um desses *Produtores* de forma a poder fornecer ao *Cliente* uma listagem dos prazos e/ou custos do serviço pretendido (fase de *Localização de Produtores* e fase de *Pedido de Orçamentos e Negociação*). O *Cliente* ao receber a melhor proposta, ou as propostas fornecidas pelo *Broker*, encomenda a que lhe for mais favorável, indicando o número da proposta e o número do orçamento. O *Broker*, ao receber a encomenda do *Cliente*, contacta o *Produtor* em causa e envia-lhe a encomenda (fase da Encomenda). O *Cliente*, a partir deste momento e até receber a encomenda nas suas instalações, pode fazer pedidos ao *Broker* de forma a saber qual o estado da encomenda em cada instante (fase do Seguimento da Encomenda).
Nas secções seguintes serão desenvolvidas, mais pormenorizadamente, cada uma das fases referidas. São também apresentados exemplos das mensagens SOAP enviadas e recebidas em cada uma dessas fases.

3.5.1 Registo dos Produtores

Cada *Produtor* necessita previamente de se encontrar registado num dos vários *servidores UDDI* (Universal Description, Discovery and Integration) de uma das companhias que disponibilizam o serviço de registo e localização de serviços WEB (Microsoft, IBM, GOOGLE, etc.) (Figura 3-9).

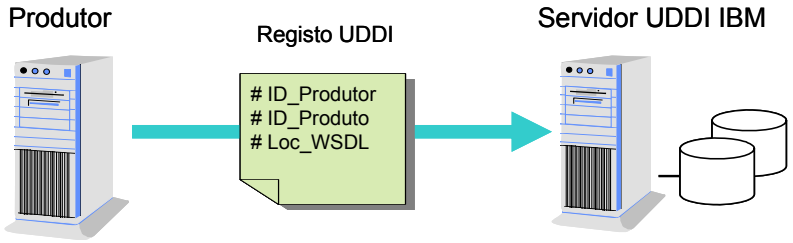


Figura 3-9 – Registo *Produtores* no Servidor UDDI

É a estes servidores que o *Broker* vai recorrer para localizar os *Produtores* que podem satisfazer os pedidos do *Cliente.* Assim, um *Produtor* que pretenda disponibilizar um serviço e mais tarde poder ser solicitado pelo *Broker*, tem que registar, num destes *servidores UDDI,* as informações respeitantes à sua empresa, produto a disponibilizar, forma e localização pela qual pode ser contactado.
O registo de serviços nos servidores UDDI da IBM pode ser feito no Browser WEB, através de uma página localizada no site da IBM ou através do PHP, usando as funções referidas no capítulo 2, mais exactamente no ponto WSDL. O endereço do servidor UDDI utilizado que

permitiu fazer uma inscrição de um serviço para realizar os testes práticos durante este trabalho foi http://uddi.ibm.com/testregistry/registry.html. Nesta página foi registado um serviço chamado "Produtores UA" onde se introduziram as informações respeitantes à entidade, assim como a descrição e a localização do serviço disponibilizado. Como confirmação do registo foi atribuído uma chave UDDI para esse serviço e uma chave *tModel* (Figura 3-10).

.

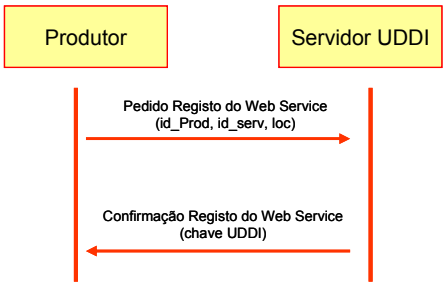


Figura 3-10 – Esquema temporal do registo UDDI

3.5.2 Fase de Pedido de Orçamentos e Negociação

Nesta secção são descritas, mais pormenorizadamente, as interacções *entre* as várias entidades quando se encontram na *fase de Pedido de Orçamentos e Negociação* (Figura 3-11).

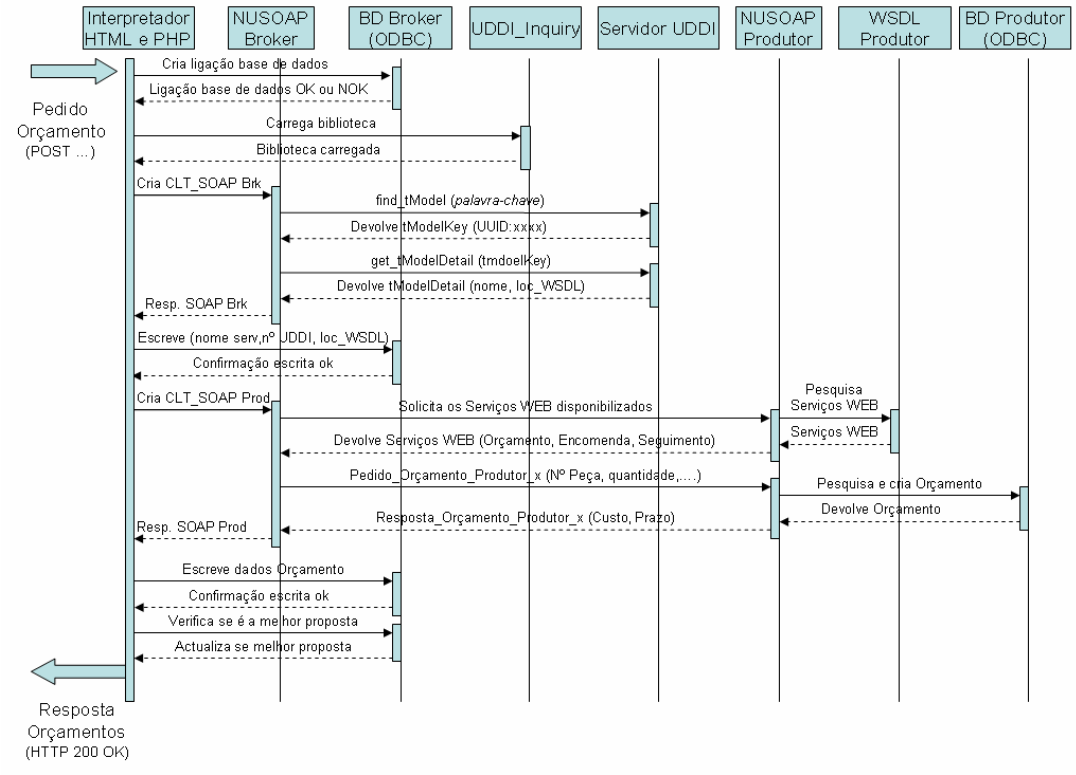


Figura 3-11 – Operações do *Broker* na fase de Pedido de Orçamentos e Negociação

Nesta fase podem ser encontradas duas das funções principais do *Broker*:

- Localização dos Produtores
- Pedido de Orçamentos e Negociação

A "*Localização de Produtores*" é a primeira função principal que o *Broker* tem que possuir. Esta função tem que permitir ao *Broker* consultar um dos vários *servidores UDDI,* com o objectivo de obter uma lista dos *Produtores* capazes de satisfazer o pedido do *Cliente,* assim bem como a sua localização na WEB*.* Quando *Broker* possuir esta lista, ou mais exactamente, já possuir os URL ou WSDL dos *Produtores*, questiona-os de forma a obter os orçamentos e prazos de entrega para o produto pretendido pelo *Cliente*. Para que isso seja possível, o *Broker* começa por pedir a cada *Produtor* o seu ficheiro WSDL com a descrição dos serviços WEB que estes disponibilizam. Aqui começa a segunda função principal do *Broker* chamada de "*Orçamento e Negociação".* O *Broker* ao receber o ficheiro WSDL de cada *Produtor* fica a saber quais são os serviços por eles disponibilizados, podendo evocá-los de forma a satisfazer os pedidos do *Cliente*. No final desta fase o *Broker* apresenta, no *Browser WEB* do *Cliente*, uma proposta com todos os orçamentos recebidos dos vários *Produtores* contactados, incluindo o orçamento mais vantajoso.

3.5.2.1 Localização Produtores

O *Cliente* ao aceder, via WEB, ao endereço URL do *Broker* http://ims.mec.ua.pt/broker/index.htm, encontra os produtos que este disponibiliza e consegue negociar. É nesta fase que o *Cliente* vai pedir ao *Broker* o pedido do melhor orçamento para um determinado produto. Assim, na página URL do *Broker* "Propostas e Orçamentos", o *Cliente* vai fornecer ao *Broker* os parâmetros de pesquisa do serviço a solicitar (peça, quantidade, material e preferência do prazo ou custo). O *Broker*, de acordo com esse pedido, deve conseguir descobrir todos os *Produtores* capazes de o realizar e solicitar-lhes um pedido de orçamento para posteriormente apresentar o melhor orçamento ao *Cliente* ("*Pedido_Orçamento*") (Figura 3-12) (ver Apêndice C.2).

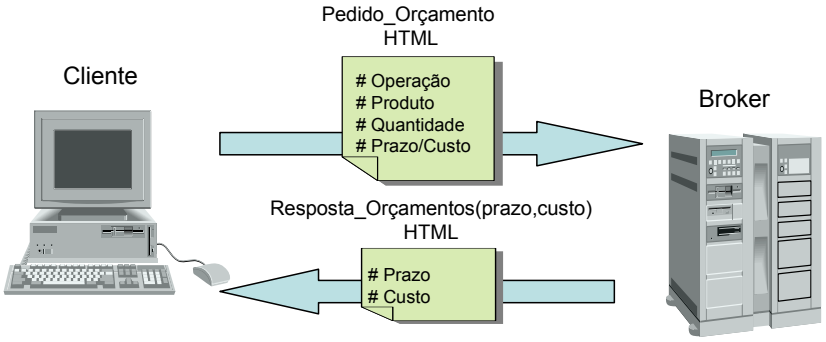


Figura 3-12 – Pedido de Orçamentos

O *Broker* ao receber estes parâmetros, pesquisa no Servidor UDDI, quais os *Produtores* aí registados e que são capazes de fornecer tal serviço. Para isso, o *Broker* vai criar e enviar, via HTTP, uma mensagem no formato SOAP para o servidor UDDI. Esta mensagem, chamada "find_tModelKey", vai conter o nome da operação ou o do produto pretendido pelo *Cliente* (ver Apêndice C.3). O Servidor UDDI ao receber esta mensagem e após realizar a pesquisa vai devolver uma outra mensagem SOAP ("Devolve_tModelKey") com os *tModelKey* que identificam cada um dos *Produtores* no servidor UDDI, como se pode ver na Figura 3-13 (ver Apêndice C.4).

.

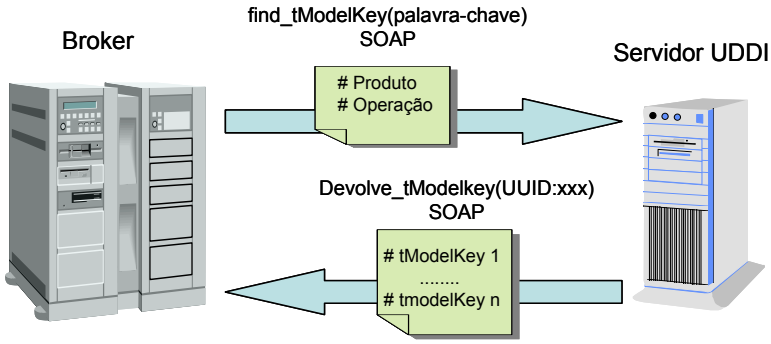


Figura 3-13 – Localização de *Produtores*

Tendo os *tModelKey,* que identificam cada um dos *Produtores* capazes de produzir o produto escolhido pelo *Cliente*, o *Broker* cria e envia uma nova mensagem ao servidor UDDI chamada "*get_tModelDetail(tModelKey)*" onde será introduzido o *tModelKey* encontrado anteriormente para cada um dos *Produtore*s (ver Apêndices C.5, C.7, C.9, C.11, C.13, C.15). Já foi referido que cada *tModelKey* identifica univocamente cada um dos serviços registados no servidor UDDI. Para cada uma das mensagens recebidas, o servidor UDDI vai responder ao *Broker* com uma mensagem SOAP "*Devolve_tModelDetail(loc_WSDL)*" (ver Apêndices C.6, C.8, C.10, C.12, C.14, C.16), contendo a localização do ficheiro WSDL onde constam os serviços disponibilizados por cada um dos *Produtores* e os parâmetros necessários para os utilizar (Figura 3-14).

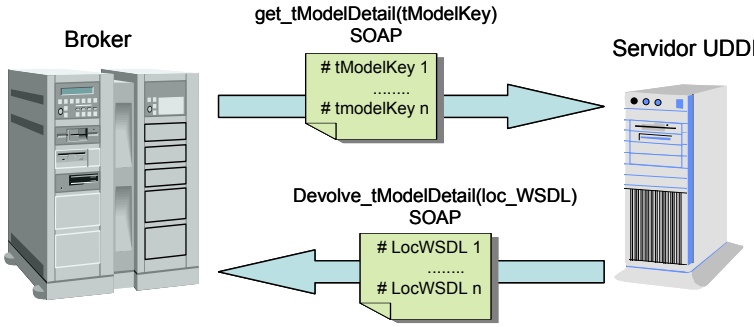


Figura 3-14 – Localização dos URL dos *Produtores* (WSDL)

Ao receber a localização dos ficheiros WSDL de cada um dos *Produtores*, o *Broker* guarda na sua base de dados "Produtores" todas as informações respeitantes a esses *Produtores*, nomeadamente o nome da operação, *tModelKey,* nome do *Produtor* e a localização do ficheiro WSDL de cada *Produtor*. Estas informações vão ser utilizadas para posteriormente contactar cada um dos *Produtores* e solicitar-lhes orçamentos, encomendas e o seguimento das mesmas (Figura 3-15).

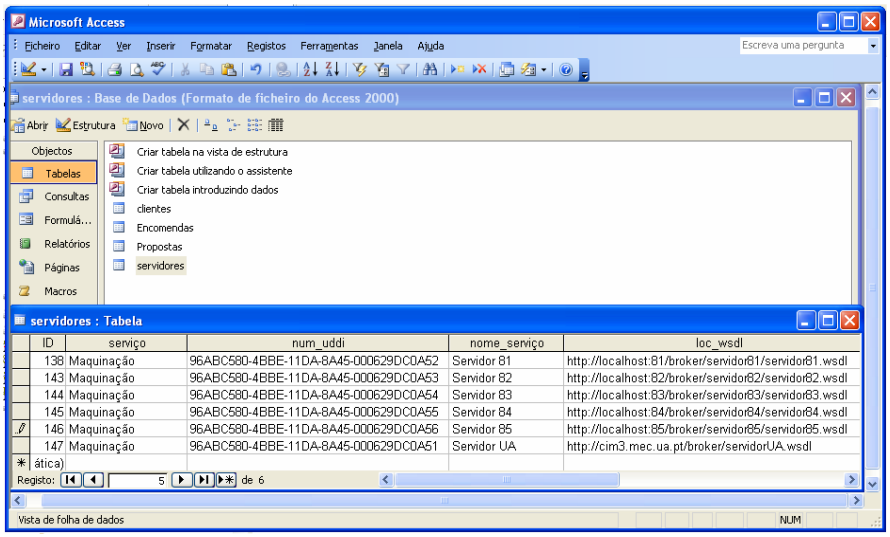


Figura 3-15 – Exemplo da base de dados do *Broker* com as localizações dos *Produtores*

Na Figura 3-16, estão representadas as fases envolvidas desde que o *Cliente* pede ao *Broker* um serviço até que este, recebe do servidor UDDI a listagem dos endereços URL, onde se encontram os ficheiros WSDL dos *Produtore*s.

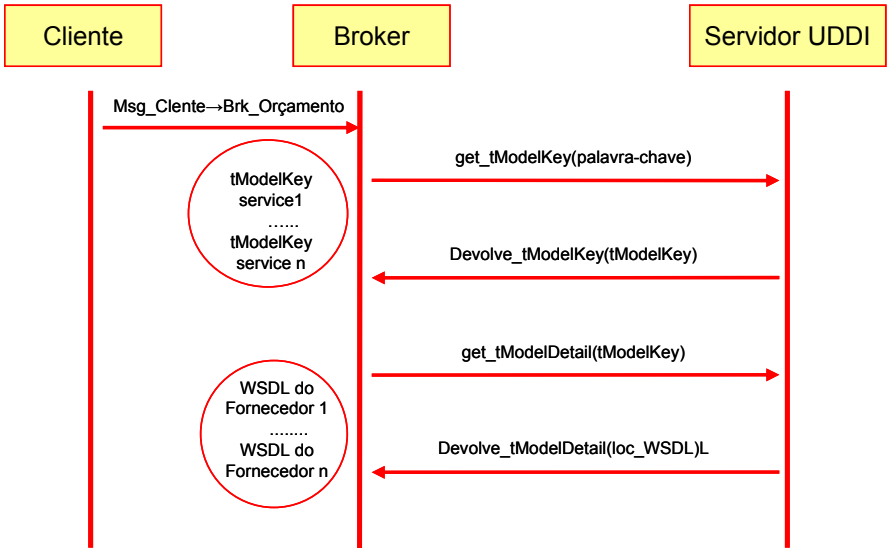


Figura 3-16 – Esquema temporal da localização de *Produtores*

3.5.2.2 Pedidos de Orçamentos e Negociação

É nesta fase que o *Broker* solicita a cada um dos *Produtores* um orçamento para o fornecimento do produto pretendido pelo *Cliente*. Os endereços WSDL de cada um dos *Produtores*, descobertos no servidor UDDI, encontram-se guardados na base de dados do *Broker* na tabela "Produtores". O *Broker* cria e envia uma mensagem SOAP ("Pedido_Orçamento_Produtor_x") para cada um dos *Produtores*, chamando o serviço "Orçamento" e passando um conjunto de dados respeitantes ao produto solicitado pelo *Cliente* (Figura 3-17) (ver Apêndices C.19, C.23, C.27, C.31, C.35, C.39).

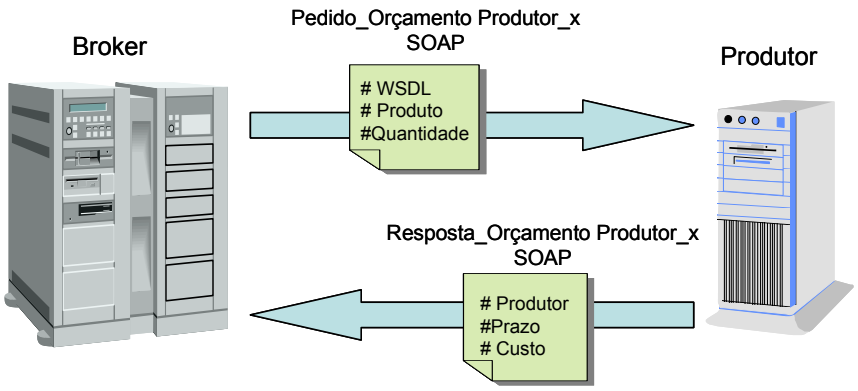


Figura 3-17 – Pedido de orçamento do *Broker* aos *Produtores*

Os *Produtores* ao receberem o pedido de orçamento consultam os dados respeitantes ao orçamento e respondem ao *Broker* com uma mensagem chamada "Resposta_Orçamento_Produtor_x", devolvendo a sua identificação, o custo e o prazo de entrega (ver Apêndices C.20, C.24, C.28, C.32, C.36, C.40). O *Broker* após receber as respostas de todos os *Produtores*, analisa todos os orçamentos e apresenta ao *Cliente* todos os orçamentos recebidos, incluindo o orçamento mais vantajoso ("Resposta_Orçamentos") (Figura 3-18).

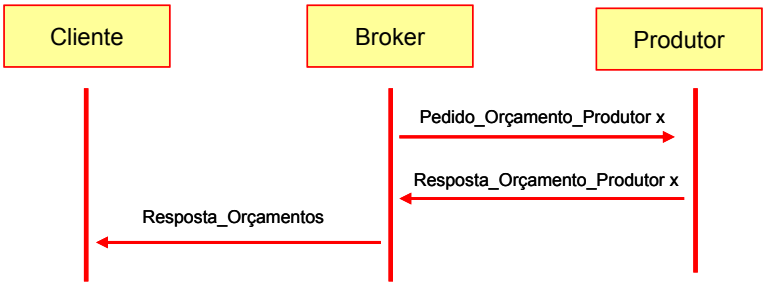


Figura 3-18 – Esquema temporal da negociação de orçamentos

Estas propostas de orçamentos ficam registadas na base de dados "Propostas" do *Broker* e são apresentadas ao *Cliente* no seu Browser WEB, como se pode ver na Figura 3-19.

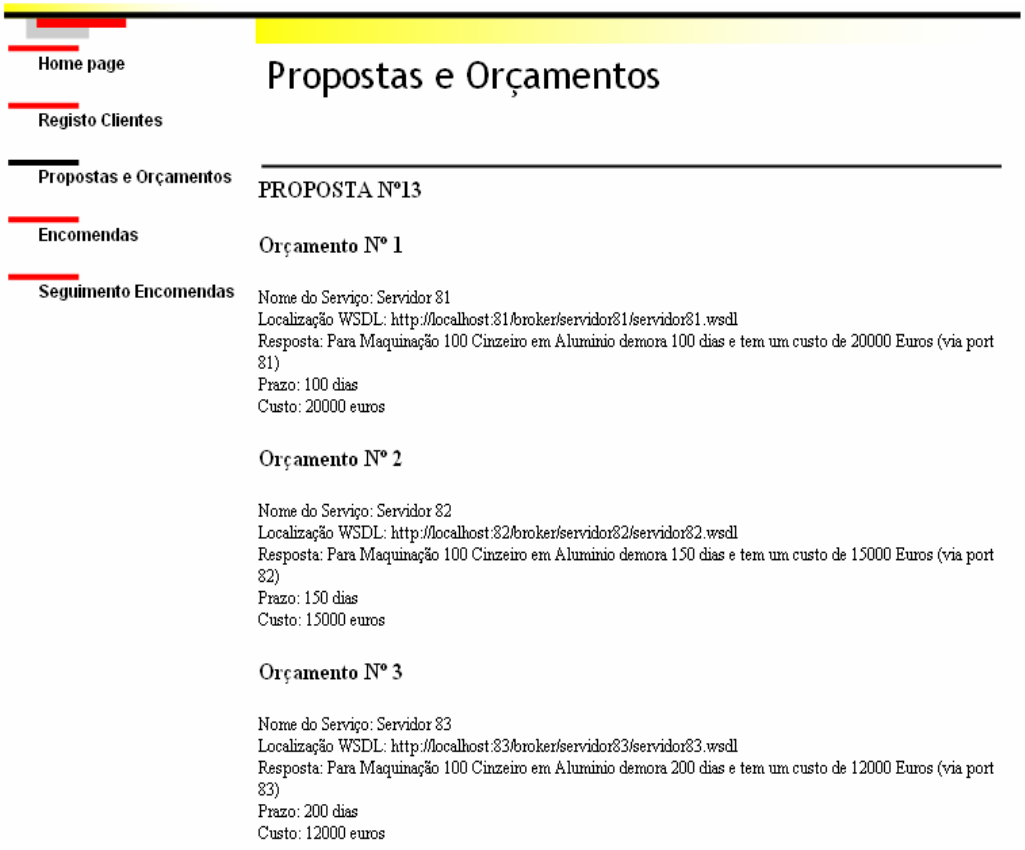


Figura 3-19 – Exemplo do Browser WEB com os orçamentos apresentados ao *Cliente*

3.5.3 Fase da Realização de Encomendas

Nesta fase o *Cliente* já recebeu no seu Browser WEB as propostas de orçamentos para o produto que pretende adquirir. É nesta fase que o *Cliente* tem a possibilidade de encomendar a o produto pretendido e para isso tem que recorrer às mensagens e fluxos de informação, mais concretamente evocar a função "Encomendas" definida no documento WSDL do *Produtor*. Esta função permite ao *Broker* interagir ou contactar os *Produtores* de forma a formalizar e realizar o pedido de produção solicitado pelo *Cliente.* A formalização ou realização dessas encomendas é feita através da escrita das informações relativas às encomendas nas bases de dados de produção dos *Produtores*. Estas mensagens e trocas de informação encontram-se ilustradas pormenorizadamente na Figura 3-20.

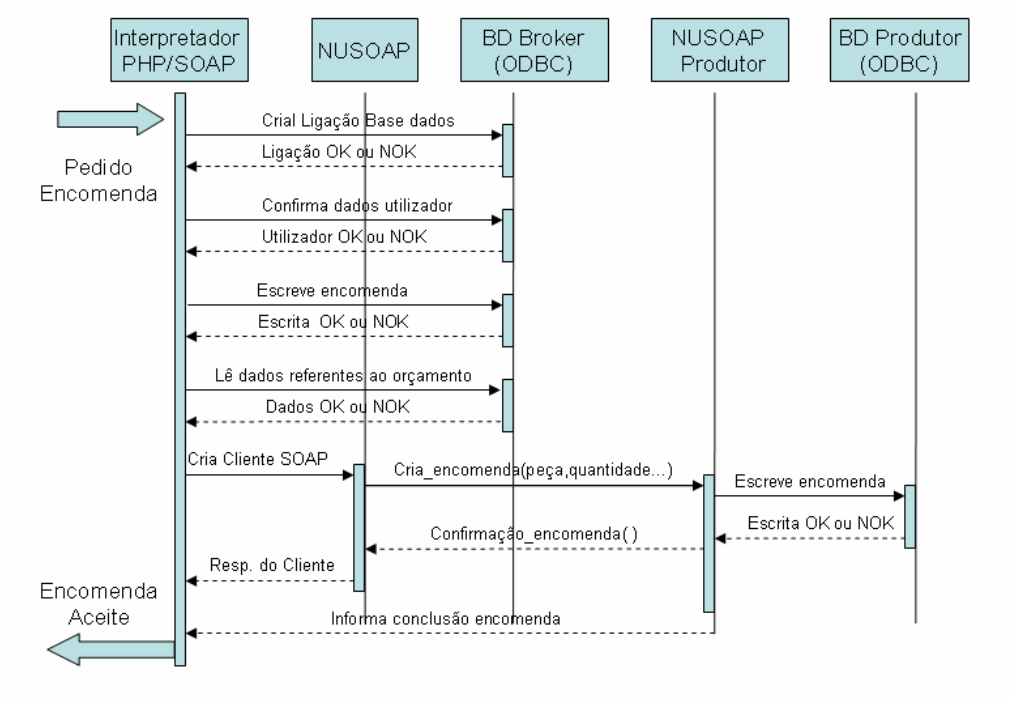


Figura 3-20 – Operações do *Broker* na fase da Encomenda

O *Cliente* ao formalizar a encomenda escolhe o orçamento que mais lhe convém, identificando-a pelo número da proposta e pelo número do orçamento. No seu Browser WEB, o *Cliente* acede à página de encomendas do *Broker*, indicando a sua identificação, proposta, orçamento e o método de pagamento escolhido ("*Pedido Encomenda*") (Figura 3-21) (ver Apêndice C.41).

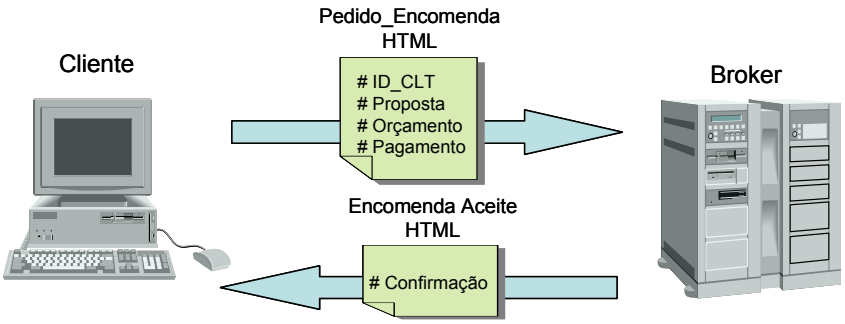


Figura 3-21 – *Cliente* encomenda um produto ao *Broker*

O *Broker* ao receber esta mensagem vai consultar a sua base de dados das "Propostas" ficando a saber o tipo do produto e serviço, a quantidade, a preferência pelo custo ou prazo, a informação e localização do ficheiro WSDL do *Produtor* previstos para essa encomenda. O *Broker* guarda na sua base de dados "Encomendas" as informações respeitantes à encomenda a solicitar ao *Produtor*. Nessa base de dados constam a identificação da encomenda, a identificação do *Cliente*, a identificação da proposta/orçamento, o tipo de pagamento, um campo de observações diversas e o estado da encomenda.

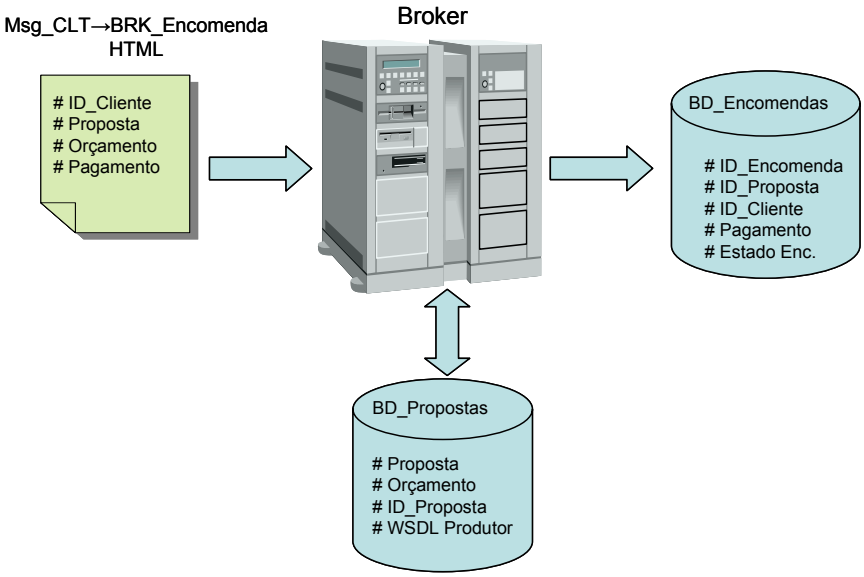


Figura 3-22 – Registo da encomenda no *Broker*

O *Broker* contacta o *Produtor* que apresentou a proposta seleccionada pelo *Cliente* através de uma mensagem SOAP "*Cria_Encomenda (peça, quantidade,…*)" (ver Apêndice C.42). Esta, criada de acordo com os serviços disponibilizados pelo ficheiro WSDL do *Produtor*, tem como parâmetros o nome do Serviço WEB a invocar ("Encomenda") e a identificação da encomenda, do *Cliente* e da proposta, conforme se pode ver na Figura 3-23. O *Broker* actualiza na sua base de dados "Encomendas" o estado da encomenda, passando-a de "Espera de Envio" para "Enviada ao *Produtor*".

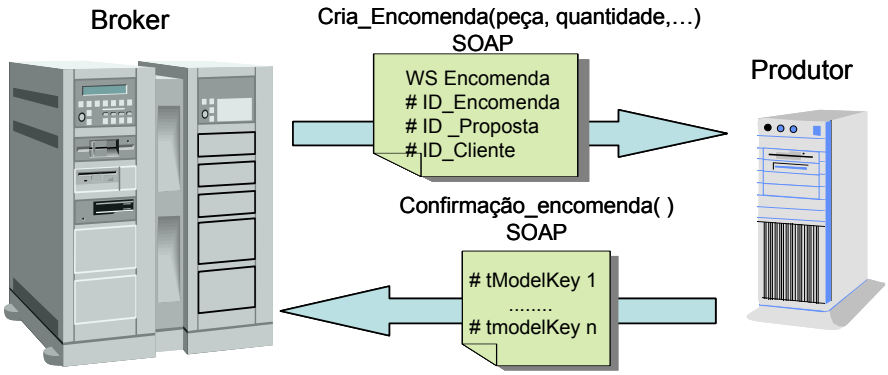


Figura 3-23 – *Broker* encomenda um produto ao *Produtor*

Os parâmetros enviados na mensagem *"Cria_Encomenda"* vão ser guardados na base de dados "Produção" existente no servidor do *Produtor* através do Serviço WEB "Encomenda", ficando a encomenda com o estado para "Fila de Espera". Seguidamente, o *Produtor* envia uma mensagem ao *Broker* a informá-lo da correcta recepção e registo da encomenda na sua base de dados de produção (*"Confirmação_encomenda()"*)(ver Apêndice C.43). Neste momento, o início de produção da encomenda fica sujeita à disponibilidade do recurso existente nas instalações do *Produtor*. O *Produtor* ao iniciar a produção da encomenda actualiza o estado passando-a de "Fila de Espera" para "Em Produção". Ao concluir a produção, o *Produtor* vai actualizar o estado dessa encomenda conferindo-lhe o estado de "Concluída" e vai preparar a encomenda final para enviar ao *Cliente Final.* Ao mesmo tempo o *Produtor* envia ao *Broker* uma mensagem a dar conta da finalização daquela encomenda ("*Conclusão_Encomenda()"*) (Figura 3-24).

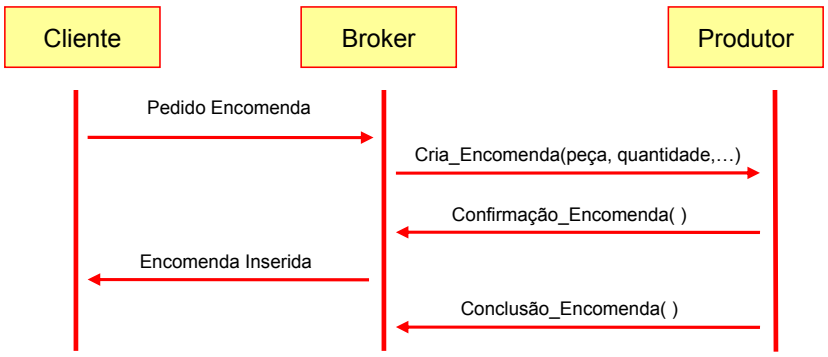


Figura 3-24 – Esquema temporal da formalização da encomenda

3.5.4 Seguimento de Encomenda

Nesta fase, a encomenda do *Cliente* já se encontra em "Fila de Espera" ou em "Produção" nas instalações do *Produtor.* Durante esta fase o *Cliente* pode contactar o *Broker* e questioná-lo sobre o estado de uma das suas encomendas em particular ("*Pedido Seguimento encomenda*") (ver Apêndices C.44, C.45). Estas mensagens e trocas de informação encontram-se ilustradas pormenorizadamente na Figura 3-27.

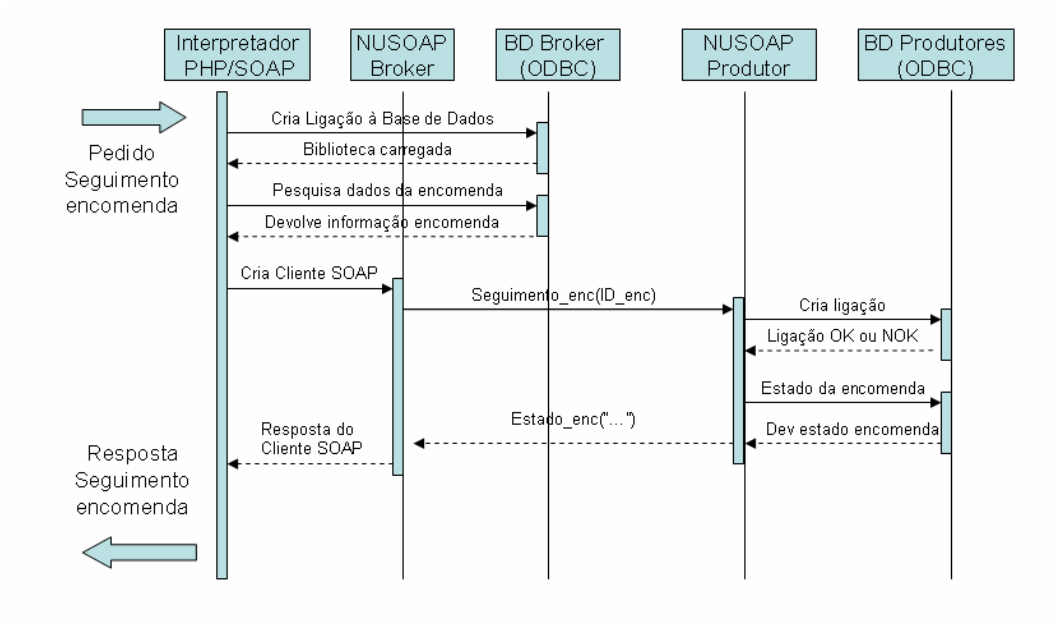


Figura 3-25 – Operações do *Broker* na fase de *Seguimento de Encomendas*

O *Broker* ao ser questionado recorre à sua quarta função chamada "*Seguimento de Encomendas*". Esta função permite interagir com os *Produtores*, de forma a saber o estado de produção de uma determinada encomenda a cada instante. No final desta fase, o *Broker* deverá apresentar, no Browser WEB do *Cliente*, a informação respeitante ao estado da encomenda.

Desde o momento em que o *Produtor* recebe o pedido de encomenda do *Broker* até ao envio da encomenda para o *Cliente*, o estado da encomenda na base de dados do *Produtor* vai sofrer actualizações consoante o desenrolar da produção. Quando o *Produtor* recebe o pedido de encomenda actualiza a sua base de dados com a nova encomenda e regista que esta se encontra no estado "Fila de Espera". Quando ela entra em produção, o *Produtor* actualiza o estado da encomenda para "Em Produção" e ao ser concluída, o estado da encomenda passa a "Concluída". O *Produtor* ao expedir a encomenda para o *Cliente* actualiza o estado desta encomenda para "Enviada ao *Cliente*".

Durante todas estas fases da encomenda, o *Broker* a pedido do *Cliente* vai consultar os *Produtores* aos quais foram entregues encomendas e solicitar-lhes o seu estado através da mensagem "*Seguimento_enc(ID_enc)"*. (ver Apêndice C.46) Esta solicitação é possível porque cada *Produtor* disponibiliza um serviço, descrito nas suas páginas WSDL, chamado "Seguimento" que ao ser acompanhado pela identificação da encomenda permite ao *Produtor* informar o *Broker* do estado da encomenda em qualquer momento, através da mensagem "*Estado_enc("…")"* (Figura 3-26).

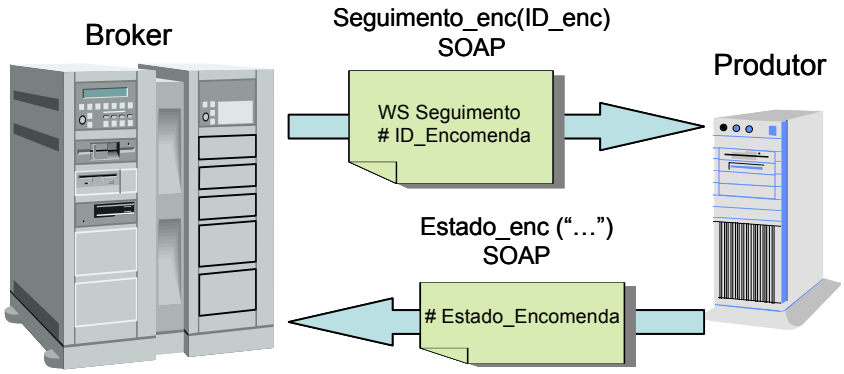


Figura 3-26 – Seguimento da produção e envio da encomenda

O *Broker* ao receber o estado da encomenda no *Produtor* solicitada pelo *Cliente* através da mensagem "*Estado_enc ("…")"* (ver Apêndice C.47) vai criar uma nova mensagem HTML e enviá-la para o Browser WEB do *Cliente*, informando-o do estado da encomenda em tempo real ("*Resposta_Seguimento_Encomendas*") (Figura 3-27).

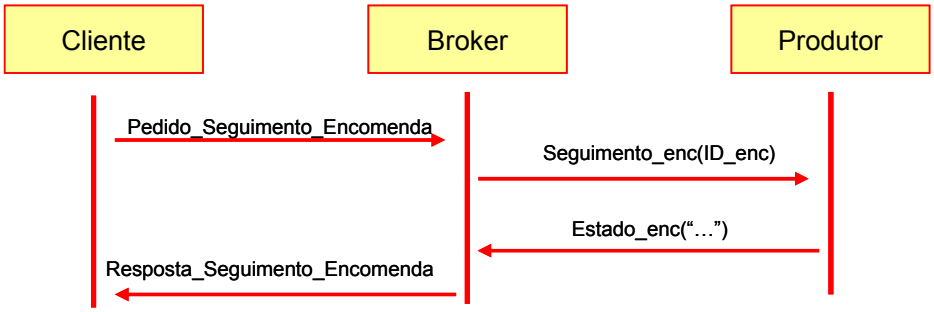


Figura 3-27 – Esquema temporal do Seguimento de Encomendas

No caso da encomenda no *Produtor* se encontrar no estado "Em Produção", o *Cliente* pode visualizar e acompanhar a produção do seu produto no Browser do seu computador ou PDA através de uma imagem obtida por uma "*Webcam" (*Figura 3-28*).*

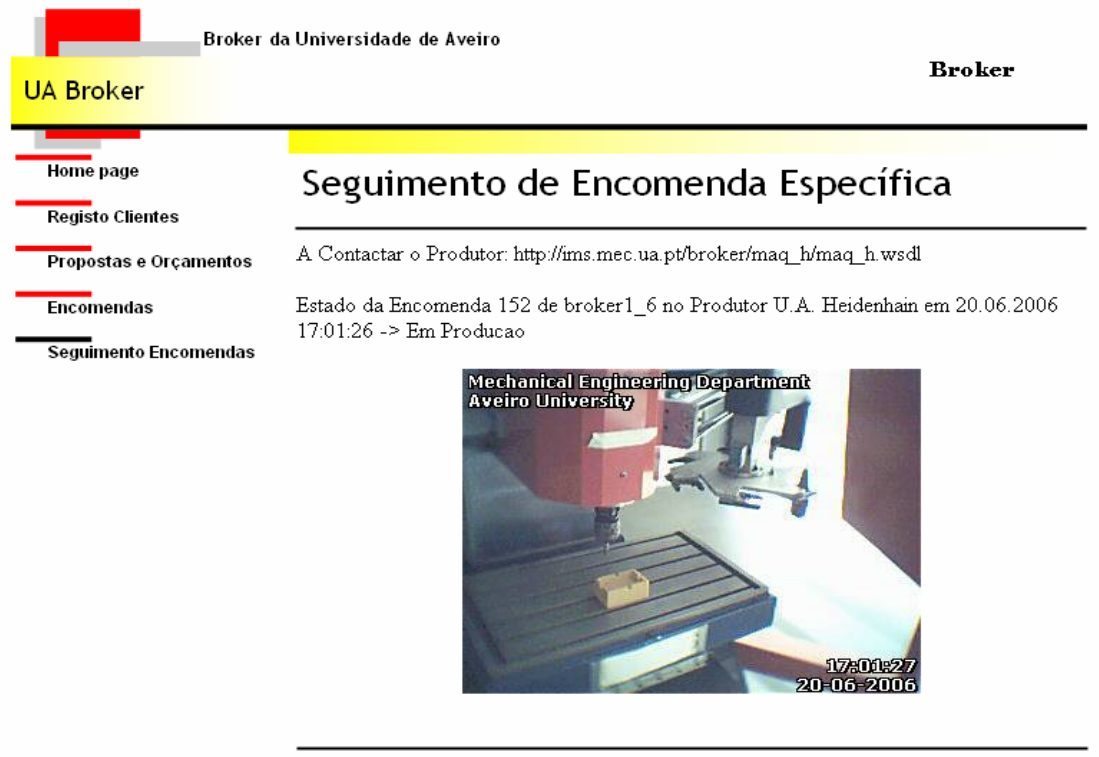


Figura 3-28 – Exemplo de uma página WEB do Seguimento de Encomendas

Capítulo 4

Resultados Práticos

4 RESULTADOS PRÁTICOS

Neste capítulo são apresentados os resultados obtidos da implementação da solução proposta, mais concretamente a medição dos tempos envolvidos durante as seguintes fases: Localização dos Produtores, Pedido de Orçamentos e Negociação, Realização de Encomendas e Seguimento de Encomendas.

Na fase da Localização dos *Produtores* pretende-se medir o tempo que o *Broker* precisa para contactar um dos servidores UDDI e obter os endereços dos vários *Produtores* (localização dos ficheiros WSDL) e guardá-los na base de dados "Fornecedores".

Na fase de Pedido de Orçamentos e Negociação pretende-se medir o tempo que o *Broker* precisa para contactar cada um dos *Produtores*, solicitar e receber um orçamento de cada um deles, analisar e ordenar todos os orçamentos recebidos consoante um critério definido pelo *Cliente*, e apresentar a melhor proposta de orçamento.

Na fase de Realização de Encomendas pretende-se medir o tempo que o *Broker* precisa para realizar e formalizar uma encomenda a um *Produtor* específico, consoante o orçamento escolhido pelo *Cliente*.

Na fase de Seguimento de Encomendas pretende-se medir o tempo que o *Broker* precisa para contactar um *Produtor*, ao qual foi atribuído a execução de um orçamento, e ficar a saber a evolução ou estado dessa encomenda no *Produtor*.

É também apresentada uma situação real, utilizada por uma empresa, com o objectivo de adquirir uma determinada peça.

Vão ser contabilizados todos os:

- Tempos para a identificação e localização dos endereços e formas de contacto dos vários *Produtores* capazes de fornecer essa peça (páginas amarelas, Internet, serviço de informações).

- Tempos necessários à redacção de documentos para a solicitação de orçamentos e formalização de encomendas a cada um dos *Produtores*.

- Tempos necessários de envio destes documentos aos *Produtores.*

- Tempos de resposta de cada um dos *Produtores*.

- Tempos para a análise e seriação dos orçamentos recebidos.

Assim sendo, este capítulo pretende dar ao leitor uma ideia dos tempos envolvidos em cada uma das fases da solução proposta e, se possível, compará-los com os tempos envolvidos no método, dito tradicional ou usual.

4.1 Tempos Obtidos na Solução Tradicional

Nesta secção apresentam-se dois pedidos reais de orçamentos a dois *Produtores Reais* para aquisição de um determinado produto. Estes *Produtores* foram contactados através dos documentos das Figura 4-1 e 4–2, enviados por fax, solicitando um orçamento para aquisição de um determinado produto (Figura 4-3), de acordo com um plano. Neste caso em concreto, apenas foram contactados dois *Produtores* por serem conhecidos e por habitualmente trabalharem para a empresa contratante. Nos pedidos, encontram-se as condições de resposta ao orçamento solicitado, bem como as condições da encomenda. O início do processo de pedido de orçamento iniciou-se no dia 17 de Maio de 2006 com a redacção destes documentos e a escolha dos *Produtores* a contactar. Como apenas foram consultados dois *Produtores*, o processo de redacção e envio dos pedidos de orçamentos não se pôde considerar um processo complexo nem moroso, no entanto, à medida que o número de *Produtores* a contactar vai aumentando, acresce também o tempo dispendido na elaboração e envio de documentos, disponibilização de meios humanos para os redigir, custos de envio do pedido das propostas aos *Produtores*.

C.A.C.I.A.

Companhia Aveirense de Componentes para a Indústria Automóvel, S.A.

C.A.C.I.A., S. A.
DIRECÇÃO DE COMPRAS INDUSTRIAIS
JOSÉ LOURENÇO
Apartado 10 – Cacia
3801-653 AVEIRO
☎ 234 30 14 24
🖷 234 30 13 32
E.MAIL manuel.lourenco@renault.com

SOPRETE
ATT: SR. ANILEIRO
☎ : 234325700
🖷 : 234325720

Cacia, 17/05/2006

N/DOSSIER- 20082073

ASSUNTO : PEDIDO DE PROPOSTA.

Exmos.Snrs.

Solicitamos nos enviem vossa melhor proposta de prêços (unitários) e prazo de entrega, para os seguintes fornecimentos :

QT.	CODIGO	DESIGNAÇÃO

PEÇAS CONF.PLANOS E LISTA ANEXA

CONDIÇÕES DE ENCOMENDA:
►Transportes e seguros á vossa conta e cargo.
► Pagamento:Transferência Bancária a 90 dias data de fim de mês de vossa factura.
►Vossa Proposta deverá ser enviada até- 3 DIAS UTEIS
►Prazo de entrega pretendido :
► A vossa proposta só poderá ser considerada válida se for devidamente referenciada, identificando nosso Nº de dossier.

Esperando vossa pronta resposta,apresentamos os nossos melhores Cumprimentos.

J.LOURENÇO

RENAULT

Figura 4-1 – Pedido de Orçamento ao *Produtor* 1

C.A.C.I.A.

Companhia Aveirense de Componentes para a Indústria Automóvel, S.A.

C.A.C.I.A., S. A.
DIRECÇÃO DE COMPRAS INDUSTRIAIS
JOSÉ LOURENÇO
Apartado 10 – Cacia
3801-653 AVEIRO
☎ 234 30 14 24
234 30 13 32
E.MAIL manuel.lourenco@renault.com

ASTRO
ATT: SR. TEIXEIRA
☎ : 234520420
: 234520429

Cacia, 17/05/2006

N/DOSSIER- 20082073

ASSUNTO : PEDIDO DE PROPOSTA.

Exmos.Snrs.

Solicitamos nos enviem vossa melhor proposta de prêços (unitários) e prazo de entrega, para os seguintes fornecimentos :

QT.	CODIGO	DESIGNAÇÃO

PEÇAS CONF.PLANOS E LISTA ANEXA

CONDIÇÕES DE ENCOMENDA:
►Transportes e seguros á vossa conta e cargo.
► Pagamento:Transferência Bancária a 90 dias data de fim de mês de vossa factura.
►Vossa Proposta deverá ser enviada até- 3 DIAS UTEIS
►Prazo de entrega pretendido :
► A vossa proposta só poderá ser considerada válida se for devidamente referenciada, identificando nosso Nº de dossier.

Esperando vossa pronta resposta,apresentamos os nossos melhores Cumprimentos.

J.LOURENÇO

RENAULT

Figura 4-2 – Pedido de Orçamento ao *Produtor* 2

Note-se que nos documentos, bem como nas condições de encomenda, encontra-se definido o prazo de resposta de três dias úteis.

Por motivos óbvios e de sigilo não foi possível apresentar o plano da peça que se pretende adquirir, mas imagine-se que a peça é a que se encontra na Figura 4-3, acompanhada com as características e dimensões pretendidas.

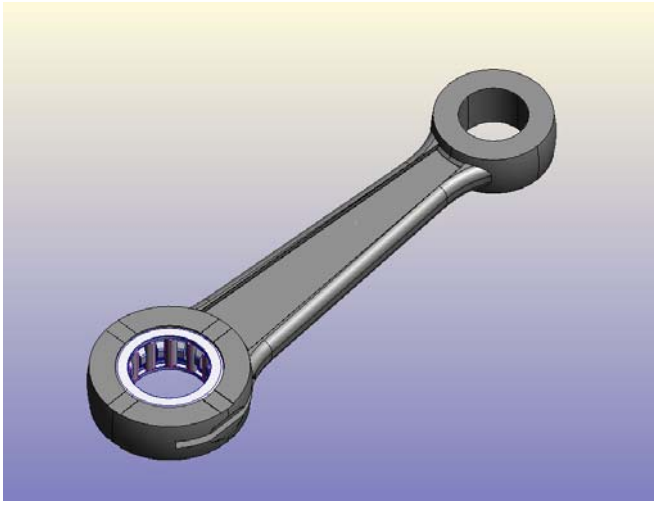


Figura 4-3 – Peça a produzir ou a adquirir

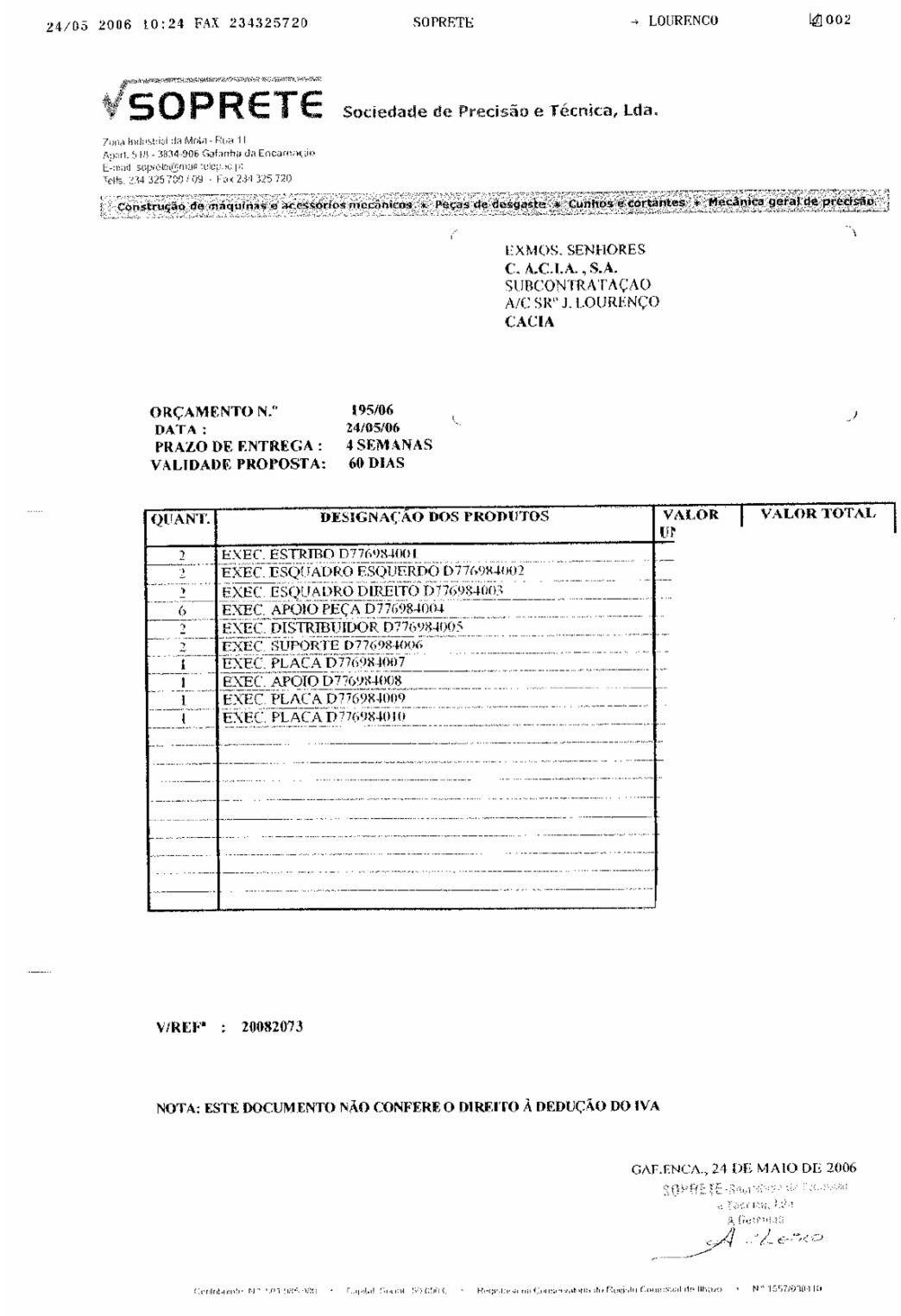
Tendo sido enviados os pedidos de proposta no dia 17 de Maio de 2006, por fax, e com um prazo de resposta de orçamento de três dias úteis, ficou-se à espera das respostas destes dois *Produtores*.

No dia 24 de Maio de 2006, o *Produtor 1* enviou, por fax, a sua melhor proposta para o fornecimento da peça que a nossa empresa pretendia adquirir. Foi necessário esperar mais dois dias para receber a proposta do *Produtor 2* e, assim, poder ser feita uma comparação e seriação das propostas destes *Produtores,* com o objectivo de escolher a melhor proposta.

Tabela 4 – Resumo do processo de pedido de orçamentos

	Data Pedido	Prazo máximo	Data Respostas	Dias em atraso
Produtor 1	17/05/2006	22/05/2006	24/05/2006	2 Dias
Produtor 2	17/05/2006	22/05/2006	26/05/2006	4 Dias

Como é óbvio, quanto maior for o número de *Produtores* contactados, maior será o tempo de espera de recepção de todas as propostas e mais demorado será o processo de seriação das mesmas. Veja-se também que o meio de comunicação (fax ou e-mail) para o pedido e resposta de orçamentos, já pode ser considerado um meio de transmissão de informação rápido, eficiente e eficaz, quando comparado, por exemplo, com a carta. Nas Figura 4-4 e Figura 4-5 encontram-se dois exemplos de respostas dos *Produtores* contactados, e mais uma vez por motivos de sigilo profissional, foram omitidos os valores em causa.



24/05 2006 10:24 FAX 234325720 SOPRETE → LOURENCO 002

SOPRETE Sociedade de Precisão e Técnica, Lda.

Zona Industrial da Mota - Rua 11
Apart. 5 (B - 3834-906 Gafanha da Encarnação
E-mail: sopretel@mail.telepac.pt
Telfs. 234 325 709 / 09 - Fax 234 325 720

Construção de máquinas e acessórios mecânicos • Peças de desgaste • Cunhos e cortantes • Mecânica geral de precisão

EXMOS. SENHORES
C. A.C.I.A. , S.A.
SUBCONTRATAÇAO
A/C SR^a J. LOURENÇO
CACIA

ORÇAMENTO N.º **195/06**
DATA : **24/05/06**
PRAZO DE ENTREGA : **4 SEMANAS**
VALIDADE PROPOSTA: **60 DIAS**

QUANT.	DESIGNAÇÃO DOS PRODUTOS	VALOR UN	VALOR TOTAL
2	EXEC. ESTRIBO D776984001		
2	EXEC. ESQUADRO ESQUERDO D776984002		
2	EXEC. ESQUADRO DIREITO D776984003		
6	EXEC. APOIO PEÇA D776984004		
2	EXEC. DISTRIBUIDOR D776984005		
2	EXEC. SUPORTE D776984006		
1	EXEC. PLACA D776984007		
1	EXEC. APOIO D776984008		
1	EXEC. PLACA D776984009		
1	EXEC. PLACA D776984010		

V/REF^a : 20082073

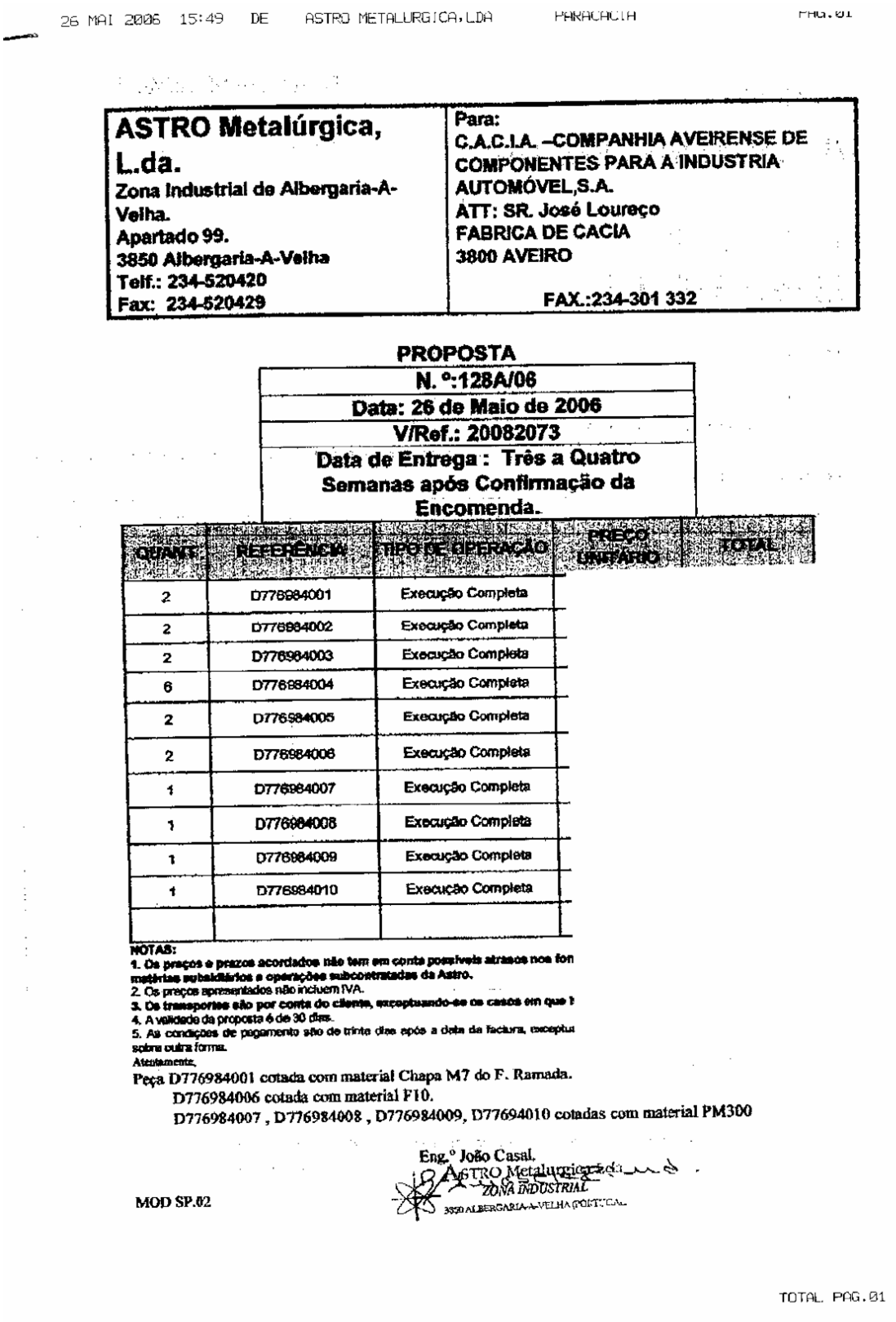
NOTA: ESTE DOCUMENTO NÃO CONFERE O DIREITO À DEDUÇÃO DO IVA

GAF.ENCA., 24 DE MAIO DE 2006

SOPRETE-Sociedade de Precisão e Técnica, Lda.
A Gerência

Contribuinte N.º 501 985 920 • Capital Social 99.760 € • Registada na Conservatória do Registo Comercial de Ílhavo • N.º 1557/970110

Figura 4-4 – Resposta de Orçamento do *Produtor* 1



26 MAI 2006 15:49 DE ASTRO METALURGICA,LDA PAIRACACIA PAG.01

ASTRO Metalúrgica, L.da.
Zona Industrial de Albergaria-A-Velha.
Apartado 99.
3850 Albergaria-A-Velha
Telf.: 234-520420
Fax: 234-520429

Para:
C.A.C.I.A. –COMPANHIA AVEIRENSE DE COMPONENTES PARA A INDUSTRIA AUTOMÓVEL,S.A.
ATT: SR. José Loureço
FABRICA DE CACIA
3800 AVEIRO
FAX.:234-301 332

PROPOSTA

N. º:128A/06
Data: 26 de Maio de 2006
V/Ref.: 20082073
Data de Entrega : Três a Quatro Semanas após Confirmação da Encomenda.

QUANT.	REFERÊNCIA	TIPO DE OPERAÇÃO	PREÇO UNITÁRIO	TOTAL
2	D776984001	Execução Completa		
2	D776984002	Execução Completa		
2	D776984003	Execução Completa		
6	D776984004	Execução Completa		
2	D776984005	Execução Completa		
2	D776984006	Execução Completa		
1	D776984007	Execução Completa		
1	D776984008	Execução Completa		
1	D776984009	Execução Completa		
1	D776984010	Execução Completa		

NOTAS:
1. Os preços e prazos acordados não tem em conta possíveis atrasos nos for
matérias subsidiários e operações subcontratadas da Astro.
2. Os preços apresentados não incluem IVA.
3. Os transportes são por conta do cliente, excepcuando-se os casos em que
4. A validade da proposta é de 30 dias.
5. As condições de pagamento são de trinta dias após a data da factura, excepto
sobre outra forma.
Atentamente,

Peça D776984001 cotada com material Chapa M7 do F. Ramada.
D776984006 cotada com material F10.
D776984007 , D776984008 , D776984009, D77694010 cotadas com material PM300

Eng.º João Casal.
ASTRO Metalurgica Lda
ZONA INDUSTRIAL
3850 ALBERGARIA-A-VELHA (PORTUGAL)

MOD SP.02

TOTAL PAG.01

Figura 4-5 – Resposta de Orçamento do *Produtor* 2

A fase de encomenda torna-se mais difícil de quantificar, visto poder estar dependente da menor ou maior rapidez da redacção de documentos que formalizem a encomenda, das burocracias e validações dos pagamentos totais ou parciais das mesmas. Seria conveniente reduzir ao máximo o tempo que todas estas acções demoram a ser realizadas. Na secção seguinte pretende-se mostrar ao leitor como se poderia utilizar o *Broker* proposto nesta dissertação para atingir esse fim.

4.2 Tempos Obtidos na Solução Proposta

Nesta secção pretende-se medir o tempo necessário para realizar o Pedido de Orçamento e Negociação para a aquisição do mesmo produto referido na secção anterior, mas desta vez, utilizando as mensagens e tecnologias de suporte adoptadas na solução proposta. Em vez de serem contactados *Produtores* conhecidos ou pesquisados manualmente pelo *Cliente*, vão ser contactados, de uma forma automática e sem intervenção do *Cliente*, todos os *Produtores* registados num dos servidores UDDI (*1 Produtor Real e 5 Produtores Simulados*), capazes de produzir a peça pretendida. *Os Produtores Simulados* encontram-se registados no servidor UDDI da IBM com o serviço “Produtores UA” e fisicamente encontram-se alojados no servidor do *Broker*. São aplicações criadas em VisualBasic que utilizam uma porta específica do Servidor WEB APACHE para comunicar com o *Broker* e têm como objectivo fornecer orçamentos. O *Produtor Real* encontra-se igualmente registado no servidor UDDI da IBM com o serviço “Produtores UA” e fisicamente encontra-se num servidor ligado a um centro de maquinação no Departamento de Mecânica da Universidade de Aveiro.
Para medir os tempos envolvidos nas várias fases foram guardadas, em variáveis do código PHP do *Broker*, as horas do início e fim de cada fase. Estas informações foram posteriormente gravadas numa base de dados, sendo assim possível saber a duração de cada fase.

O *Cliente* ao preencher a página inicial de *Pedido de Orçamentos e Propostas* no seu Browser WEB, enviou ao *Broker* o nome do serviço “Produtores UA”. O *Broker* ao receber este pedido, guardou a hora que caracterizou o início do processo de pesquisa de *Produtores* e começou a pesquisar no Servidor UDDI todos os *Produtores* que disponibilizavam este serviço. Ao serem identificados, foram guardados, na base de dados “Fornecedores” do *Broker*, os endereços das localizações onde se encontravam os ficheiros WSDL que continham os serviços disponibilizados por esses *Produtores*. A pesquisa dos *Produtores* no servidor UDDI terminou quando foram identificadas, analisadas e guardadas, em base de dados, todas as localizações dos ficheiros WSDL dos vários *Produtores*. Nesse momento foi guardada a hora que caracterizou o fim do processo de pesquisa de *Produtores*.

Tendo estas localizações na base de dados, o *Broker* guardou a hora de início do processo de solicitação de orçamento e contactou cada um dos *Produtores (Real e Simulados)* através dos seus endereços WSDL, invocando o serviço "Orçamentos" com os respectivos parâmetros que permitiram ao *Produtor* saber exactamente a peça à qual deveria fornecer um orçamento. O *Produtor Real* consultou as suas bases de dados "Peças" e "Produção" onde constavam, respectivamente, os preços unitários para as peças conhecidas e propostas pelo *Broker* e a disponibilidade do seu recurso para produzir essa peça, enquanto que os *Produtores Simulados* forneceram prazos e custos aleatórios. Os *Produtores* criaram então uma mensagem onde constavam o prazo e o custo para a produção da peça e enviaram-na ao *Broker*. De referir que todo este processo foi realizado automaticamente e sem interferência humana. O *Broker* ficou à espera da resposta dos orçamentos de cada um dos *Produtores* e à medida que os foi recebendo, guardou-os numa base de dados "Propostas" e ordenou-os automaticamente consoante o critério definido pelo *Cliente*. O *Broker* após ter ordenado todos os orçamentos, apresentou-os ao *Cliente*, incluindo também o mais vantajoso, guardando a hora que caracterizou o fim da fase de solicitação de orçamento. Nesta fase, os tempos medidos, utilizando a solução proposta, para contactar os cinco *Produtores* disponíveis rondou os 2 segundos, podendo futuramente sofrer algumas variações, devidas ao tráfego de rede aquando do contacto ao *Servidor Real*.

A fase de *Formalização e Realização de Encomendas* começou quando o *Cliente* preencheu os campos da página "Encomendas" do Browser WEB do *Broker* e validou o seu envio, sendo registada a hora da realização da encomenda. O Broker escreveu a encomenda na base de dados de produção do *Produtor* e este devolveu uma mensagem a confirmar a boa recepção dos dados. Nesse momento foi memorizada e guardada a hora que caracterizou o fim do processo de encomenda. Não podendo comparar os tempos da solução proposta com os do método tradicional, apresentam-se os tempos medidos na fase de *Formalização e Realização de Encomendas*. Verificou-se que o *Broker* necessita apenas de 1 a 2 segundos para escrever na base de dados do *Produtor* que forneceu a proposta escolhida pelo *Cliente*.

Capítulo 5

Conclusões

5 CONSIDERAÇÕES FINAIS, CONCLUSÕES E TRABALHO FUTURO

Neste capítulo, em primeiro lugar, vou apresentar as minhas considerações finais e aspectos que considerei relevantes do estudo, implementação e escrita desta dissertação. Em seguida farei uma apresentação resumida do objectivo do trabalho e da solução proposta, bem como as conclusões que dela foi possível obter. Por fim serão ainda focados alguns assuntos que foram abordados neste trabalho, de uma forma conceptual e genérica é certo, mas que poderiam ser alvo de futuros estudos ou evoluções, podendo enriquecer ou complementar este estudo que hoje aqui se apresenta.

5.1 Considerações Finais

A Internet vem-se afirmando, a cada dia que passa, um meio privilegiado de trocar e aceder todo o tipo de informação de uma forma rápida e eficaz. Para mim, com formação em Engenharia Mecatrónica, entrar numa área tão abrangente e complexa como o campo das novas tecnologias baseadas na Internet não foi fácil e exigiu muita dedicação, investigação, estudo e horas de programação. Para além destas, ainda foi bastante gratificante e motivador adquirir conhecimentos na área das programações e comunicações industriais, o que me facultou algumas ferramentas de trabalho, tanto a nível pessoal como profissional. A solução a adoptar para a implementação de uma plataforma WEB que permitisse a Localização, Negociação e Contratação de Serviços na WEB nunca esteve claramente definida e sempre foi alvo de constantes alterações e evoluções. Antes mesmo de começar a pensar o modo de funcionamento ou implementação do mecanismo, que de uma forma automática pudesse Localizar, Negociar e Contratar Serviços na WEB, foram realizados alguns estudos que se consideraram importantes para uma correcta escolha das tecnologias a adoptar. Alguns desses estudos deram origem a artigos que foram submetidos e apresentados em conferências internacionais da especialidade. Podem salientar-se:

O artigo "*WEB Servers Performance Assessment*", apresentado na Florida, Orlando, U.S.A., na 2ª Conferência Internacional da CITSA (Cybernetics and Information Technologies, Systems and Applications) e que pretendeu mostrar os desempenhos e diferenças existentes entre os dois maiores Servidores WEB (Apache e o IIS) que actualmente são usados na criação de Plataformas WEB.

O artigo "*WEB Enabled CNC Milling Machine Control*", apresentado em Saint-Etienne, França, no "12th IFAC Symposium on Information Control Problems in Manufacturing (INCOM2006)" e pretendeu mostrar a implementação de uma infra-estrutura capaz de disponibilizar, na WEB, um centro de maquinação permitindo a simulação e maquinação de peças.

O artigo "*The OPC and the Shop Floor Control*", apresentado em Saint-Etienne, França, no "12th IFAC Symposium on Information Control Problems in Manufacturing (INCOM2006)" que pretendeu apresentar e comparar algumas soluções de comunicação (ActiveX, DDE, OPC) utilizadas em plataformas computacionais que visam monitorizar e controlar processos industriais remotamente.

O artigo "*The OPC in Warehouse Management System*", apresentado em Saint-Etienne, França, no "12th IFAC Symposium on Information Control Problems in Manufacturing (INCOM2006)" que pretendeu mostrar um exemplo de uma aplicação de traçabilidade, desenvolvida e aplicada num recurso fabril de uma empresa da região, recorrendo ao OPC.

À medida que os objectivos iam sendo atingidos iam sempre surgindo novas ideias, que sempre que possível foram tidas em conta e implementadas, de forma a melhorar ou fiabilizar as funcionalidades, conceptuais ou funcionais, da plataforma proposta.

Como solução final, obteve-se uma plataforma WEB robusta, de simples utilização, interactiva, flexível, disponível 24 horas por dia, 365 dias por ano, capaz de uma forma automática e autónoma (sem intervenção humana) localizar em todo o mundo os produtores capazes de produzir uma determinada peça, pedir-lhes orçamentos de uma forma imediata e sem necessidade de redacção de documentos para os formalizar, negociar e avaliar a melhor proposta, formalizar, fazer o pagamento e seguimento das encomendas através da WEB. A plataforma implementada permite ainda a gestão e monitorização remota da produção de um centro de maquinação existente no Departamento de Mecânica da Universidade de Aveiro. Consciente de que ainda há muito trabalho, melhorias e evoluções que podem ser realizadas, pretende-se com este trabalho, lançar as bases para o desenvolvimento e fiabilização de um sistema ainda mais disponível, complexo, inteligente, autónomo, flexível e sobretudo mais amigável com o utilizador.

5.2 Conclusões

Nesta dissertação foi apresentado um mecanismo automático e autónomo de localização de Serviços WEB, baseado em especificações e protocolos WEB, como forma de reduzir os tempos e custos de negociação e contratação durante o processo de aquisição de um determinado bem ou serviço. Este mecanismo, ao qual foi dado o nome de *Broker*, utiliza o protocolo HTTP e as especificações SOAP, WSDL e UDDI para localizar e interagir com Serviços WEB.

No primeiro capítulo pretendeu-se apresentar o contexto do problema da subcontratação de bens ou serviços sem recorrer ao mecanismo proposto nesta dissertação. Pôde-se concluir que existem custos importantes que estão directamente relacionados com um incompleto, ultrapassado e demorado processo de localização de entidades fornecedoras de serviços e com todos os procedimentos burocráticos necessários para a formalização e pedido de orçamentos e encomendas. Foi também apresentada uma solução genérica onde se pretendeu caracterizar os modos de funcionamento, as características conceptuais e os pontos mais importantes para uma posterior implementação.

No segundo capítulo pretendeu-se apresentar o conjunto de tecnologias de suporte que permitissem implementar o *Broker*. Assim, foram descritas as tecnologias como o DCE, CORBA, OPC e Serviços WEB (SOAP, UDDI, WSDL). Também foram feitas algumas considerações sobre estas tecnologias no que diz respeito à facilidade, flexibilidade e rapidez de implementação, assim como as suas potencialidades nos processos de registo, localização e definição de Serviços WEB. Ainda neste capítulo foram contextualizados trabalhos de outros autores sobre estas tecnologias de suporte. Procurou mostrar-se como elas podem ser e são utilizadas noutros contextos e ramos da Ciência e Tecnologia.

No terceiro capítulo apresentou-se a solução proposta para responder aos problemas enumerados no primeiro capítulo. Descreveu-se a tecnologia de suporte seleccionada, mais exactamente os WEB Services, as entidades envolvidas e suas implementações, fluxos de informação entre elas, definições dos Serviços WEB e mensagens propostas para as utilizar ou invocar.

No quarto capítulo pretendeu-se fazer um levantamento e comparação dos custos e tempos envolvidos durante a solicitação de orçamentos para a produção de uma peça, utilizando os meios tradicionais e a Tecnologia de Suporte implementada na solução proposta.

O aspecto mais importante e cerne do problema inicialmente apresentado, centrou-se na tentativa de reduzir ou mesmo eliminar os tempos e custos inerentes à localização e subcontratação de serviços ou produtos. Mostrou-se que a utilização do *Broker* proposto torna a localização e a subcontratação de produtos ou serviços num processo mais abrangente, simples, rápido e de negociação automática.

Abrangente porque, recorrendo à utilização das "Páginas Amarelas" UDDI, permite ao *Broker* descobrir fornecedores de serviços a nível mundial, não se restringindo à localização de um conjunto limitado e conhecido de fornecedores. O facto de localizar um maior número de fornecedores de serviços implica um maior número de propostas, logo uma maior probabilidade de existir uma proposta economicamente mais favorável para a aquisição de um determinado produto ou serviço.

Simples, rápida e de negociação automática porque o *Broker* ao localizar todos os *Produtores* que permitem a aquisição de um determinado bem ou serviço, vai contactá-los e pedir-lhes automaticamente a sua melhor proposta para o fornecimento desse bem ou serviço. Este método elimina todo o processo de realização de contactos telefónicos e de escrita de documentos para solicitar orçamentos (para cada um dos *Produtores* a contactar). Por ser um processo autónomo, automático e em tempo real, reduzem-se substancialmente os custos e os tempos de resposta dos pedidos de orçamentos aos *Produtores*. Em primeiro lugar, porque ao não recorrer ao tradicional envio da solicitação de orçamentos através de correio reduzem-se os custos inerentes a esse envio e reduz-se significativamente o tempo de recepção da solicitação do orçamento no *Produtor*. Em segundo lugar, porque a avaliação e resposta desse orçamento não depende da disponibilidade de uma pessoa. Um *Produtor*, sempre disponível na WEB, ao ser confrontado pelo *Broker* com um pedido de orçamento, pode a cada instante consultar as suas bases de dados "Peças", onde constam os custos e tempos de maquinação, e "Produção", onde constam as disponibilidades dos recursos fabris, e fornecer automaticamente um orçamento. O *Broker* fica a saber, quase instantaneamente, todos os orçamentos de cada um dos *Produtores*. Neste momento, o *Broker* disponibiliza outra funcionalidade, que consiste na Negociação e Avaliação automática da melhor proposta que pode poupar muito trabalho e dinheiro na aquisição de um produto. O *Broker* ao receber as propostas de cada um dos *Produtores*, vai compará-las entre si, segundo um critério de seriação anteriormente definido pela entidade contratante (prazo de entrega ou custo). Em seguida, vai ordená-las automaticamente, apresentando ao *Cliente* todas as propostas recebidas, incluindo a economicamente ou temporalmente mais vantajosa.

Como se pode constatar, a utilização do *Broker* apresenta reduções significativas nos tempos e custos de localização e negociação, desde o instante em que se pretende adquirir um bem ou serviço através do pedido de orçamento, até ao momento em que se obtém a melhor proposta.

Uma outra potencialidade do *Broker* que se pretendeu apresentar neste trabalho foi a capacidade de formalização e realização de encomendas baseadas nos orçamentos anteriormente analisados e negociados, a qual apresenta vantagens e facilidades em relação aos processos tradicionais. Por estar disponível na WEB permite desde logo o pagamento electrónico automático, eliminando todas as burocracias e custos das transferências bancárias e a perda ou extravio de cheques ou dinheiro enviados por correio, etc. Torna-se assim num processo rápido e seguro de pagamento. Permite ainda a formalização e realização de encomendas em qualquer parte do mundo, não sendo necessário perder tempo na elaboração e envio de documentos para tal fim. Todo o processo pode ser feito em tempo real preenchendo um formulário no Browser WEB que pode ser acedido através de um PDA, PC ou até mesmo num telemóvel com acesso à Internet. A entidade contratante (*Cliente*) não tem que se preocupar em saber quais são os *Produtores* para cada uma das suas encomendas, passando a interagir apenas com o *Broker*. Assim, o relacionamento com os *Produtores* passa a ser transparente no que diz respeito às suas localizações, documentos a enviar, métodos de pagamento e outras burocracias exigidas.

Finalmente ainda foi possível apresentar o seguimento de encomendas, mais uma das potencialidades do *Broker*. Independentemente do *Produtor*, da quantidade de encomendas e do instante, é possível à entidade contratante saber o estado da sua encomenda no *Produtor* desde que é realizada até a receber nas suas instalações. Esta vantagem torna-se bastante importante quando é decisivo saber exactamente a evolução ou o tempo estimado para a recepção de um produto encomendado.

De acordo com os estudos apresentados nesta dissertação, pode-se perceber que a utilização de WEB Services, como tecnologia de suporte adoptada, vem eliminar problemas de compatibilidade entre aplicações que pretendem trocar informações ou comunicar entre si em sistemas computacionais distribuídos. As aplicações que utilizam Serviços WEB, por seguirem os padrões definidos pelo W3C, não põem nenhum tipo de restrição em relação às linguagens de programação ou plataformas usadas na sua criação. Assim, optou-se por utilizar WEB Services na implementação do *Broker*. O UDDI permite a descoberta de fornecedores de serviços a nível mundial, o WSDL permite descrever os Serviços WEB disponibilizados e a forma de interagir com cada um dos fornecedores e o SOAP permite estruturar as mensagens a enviar via HTTP ou por outro protocolo de comunicação, de forma a poderem ser interpretadas e processadas automaticamente. Um factor que também influenciou na escolha dos WEB Services foi o facto de permitirem, de uma forma simples e sem grandes alterações à estrutura existente, apagar, modificar e adicionar novas funcionalidades.

5.3 Trabalhos Futuros

Uma das limitações deste trabalho centra-se na capacidade de o *Broker* contactar *Produtores* que não constem na sua lista ou que não disponham de Serviços WEB que o *Broker* é capaz e sabe implementar. Idealmente, o *Broker* ao pesquisar os *Produtores* por uma palavra-chave no servidor UDDI, deveria possuir todos os mecanismos que possibilitassem uma completa interacção entre eles. Apesar de conhecer os serviços disponibilizados pelos *Produtores*, o *Broker* deveria conseguir adaptar-se a cada um deles, de uma forma específica e individualizada. Por exemplo, o *Broker* ao pretender solicitar um orçamento sabe que tem que utilizar um serviço "Orçamento", existente em todos os *Produtores* que ele conhece (Real e/ou Simulados). Mas, e se o *Broker* contactar um *Produtor* que possui um serviço que forneça orçamentos, mas que em vez de se chamar "Orçamento" de chamar "Orçamentar"? E se o *Produtor* não disponibilizar um serviço que forneça orçamentos? Como e que ele irá reagir? Se o *Broker* chamar o serviço "Orçamento", o *Produtor* ao receber este o pedido não o vai conseguir compreender nem conseguir interpretar, visto só conhecer o serviço "Orçamentar". Assim, ao longo deste trabalho, os *Produtores* contactados (Reais e Simulados) possuem todos os mesmos serviços, serviços estes que são conhecidos e sabem como ser invocados pelo *Broker*. Desta forma, o *Broker* apenas consegue interagir com os *Produtores* que possuem os serviços que ele invoca, podendo este sistema de interacção comparar-se com os conhecidos negócios de *"franchising".* Um trabalho futuro possível passará por tornar mais transparente o processo de interacção com os *Produtores* que possuam serviços não conhecidos ou programados no *Broker*.

Uma outra limitação do trabalho apresentado centra-se no tipo de produtos disponibilizados e que são capazes de ser tratados pelo *Broker*. Um *Cliente* ao aceder à página WEB do *Broker*, vê-se obrigado de adquirir apenas produtos disponibilizados pelo mesmo, não sendo possível ao *Cliente* solicitar orçamentos e encomendar produtos do seu interesse. Os *Produtores* apenas conseguem fornecer orçamentos e produzir as peças que se encontram na página WEB do *Broker*, visto apenas possuírem os programas peça necessários para os produzirem nos seus recursos fabris (centros de maquinação). Seria interessante que os *Produtores* possuíssem a capacidade de analisar, autonomamente e automaticamente, um plano da peça ou um programa peça enviado pelo *Cliente* durante o contacto com o *Broker* e a partir dele, conseguirem fornecer um orçamento para a sua produção. Um trabalho futuro possível passará por possibilitar aos *Produtores* a capacidade de orçamentar um produto solicitado por um *Cliente* recorrendo, por exemplo, à simulação programa peça e atribuindo um preço/hora máquina.

Um outro aspecto abordado na solução proposta, mas que não foi dada a devida atenção e desenvolvimento por estar fora do contexto do tema deste trabalho foi a questão dos pagamentos electrónicos através da WEB. Trabalhos futuros poderão ser desenvolvidos abordando as seguranças e os meios de pagamentos na WEB.

Bibliografia

6 BIBLIOGRAFIA

[ACCESS 2006]
Microsoft Office Access 2003, 2006
http://office.microsoft.com/en-us/assistance/CH790018001033.aspx

[Alvarinhas 2005]
Hugo Alvarinhas, André Quinta, José Santos, "Web Servers Performance Assessment", 2nd International Conference on Cybernetics and Information Tecnologies Sysytems and Applications (CITSA2005), Orlando, USA, 2005

[ANACOM 2006]
ANACOM – Autoridade Nacional de Comunicações
Web site: http://www.icp.pt/

[APACHE 2005]
The Apache Software Foundation, 2005
Web site: http://apache.org

[CORBA 2004]
Overview of CORBA, 2004
Web site: http://www.cs.wustl.edu/~schmidt/corba-overview.html

[Curbera 2002]
Francisco Curbera, Matthew Duftler, Rania Khalaf, William Nagy, Nirmal Mukhi, Sanjiva Weerawarana ,"Unravelling the Web Services - An Introduction to SOAP, WSDL and UDDI", IEEE Internet computing nº.2, pp 86-93, Março/Abril 2002.

[DCE 1996]
DCE – Some DCE Papers Available online, 1996
Web site: http://www.opengroup.org/dce/info/papers/

[DCE 2001]
DCE – Alive and Well in Data-Center Near you, 2001
Web site: http://www.ukuug.org/events/linux2001/papers/html/CBenson/t1.html

[DCE 2005a]
Distributed Computing Environment, 2005
Web site: http://www.opengroup.org/dce/

[DCE 2005b]
Carnegie Mellon Software Engineering Institute, "Distributed Computing Environment – Software Technology Roadmap), 2005
http://www.sei.cmu.edu/str/descriptions/dce.html

[DEV 2005]
DEV – Creating your own Private UDDI Registry, 2005
Web site: http://devx.com/java/article/21390/1954?pf=true

[Dias 2006]
Nuno Dias, Hugo Alvarinhas, André Quintã, José Santos, "WEB Enabled CNC Milling Machine Control", Incom2006 – 12th IFAC Symposium on Information Control Problems in Manufacturing, Saint-Etienne, France, 2006

[Forrester 2006]
Forrester Research – Helping Business Thrive on Technology Change
Web site: http://forrester.com

[Fraunhofer 2006]
Fraunhofer – Institute Integrierte Schaltungen
Web site: http://www.iis.fraunhofer.de/ec/middle/index.html

[Gartner 2006]
Gartner Group
Web site: http://www.gartner.com/

[Gomes 2003]
António Filipe Nunes Gomes, "WEBGRAF – Aplicação Web para Execução de GRAFCETS e Redes de Petri em Controladores Lógicos Programáveis", Dissertação de Mestrado em Automação, Instrumentação e Controlo da Faculdade de Engenharia do Porto, Porto, Portugal, 2003

[IBM 2006]
IBM Software – Solutions – Web Services – UDDI, 2006
Web site: http://www-3.ibm.com/services/uddi

[James&Smit 2005]
"Service-Oriented Paradigms in Industrial Automation"
François Jammes e Harm Smit
IEEE Transactions on Industrial Informatics,
Volume 1, Issue 1, Feb. 2005 Page(s):62 - 70

[Java 2006]
Java Technology and Web Services, 2006
Web site: http://java.sun.com/webservices/index.jsp

[Lockhart 1994]
Harold W. Lockhart, "OSF DCE Guide to Developing Distributed Applications", MacGraw-Hill, 1994

[Lopes 2004]
Carlos J. Feijó Lopes, José Carlos Ramalho, "Web Services: Metodologia de Desenvolvimento", 2004

[Lopes 2005]
Carlos Lopes, José Ramalho, "Web Services – Aplicações Distribuídas sobre Protocolos Internet", FCA, 2005

[Menéndez 2002]
Andrés Ignácio Martinez Menéndez, "Uma Ferramenta de Apoio ao Desenvolvimento de Web Services", Dissertação de Mestrado em Informática da Universidade Federal de Campiona Grande, capítulo 2, pag19, Campina Grande, Brasil 2002.

[Merino 2004]
Yesenia Marisol Vigil Merino, "Web Service and Architecture For Accessing Fieldbus Systems), Master Thesis of Technical University Hamburg, Hamburg, Alemanha, 2004

[NuSOAP 2006]
Dietrich, Ganx4 NuSOAP, Junho 2006.
Web site: http://dietrich.ganx4.com/blog/?cat=41

[OMG 2006a]
Object Management Group OMG, 2006
Web site: http://www.omg.org

[OMG 2006b]
Object Management Group OMG – History of CORBA, 2006
Web site: http://www.omg.org/gettingstarted/history_of_corba.htm

[PHP 2006]
Site official do PHP – Personal Home Page, 2006
Web site: http://www.php.net

[Potts 2003]
Stephen Potts, Mike Kopack, "Aprenda em 24 Horas Web Services", Campus, 2003

[MSDN 2006]
Microsoft MSDN, 2006
Web site: http://msdn1.microsoft.com/en-us/default.aspx

[Nunes 2004]
Mauro Nunes, Henrique O'Neill, "*Fundamental de UML*", FCA, 2004

[Ribeiro 2005]
Danielle Corrêa Ribeiro, Elizabeth Mª Martinho da Silva, Francisco A. S. Júnior, Thatiane de Oliveira Rosa, Madianita Bogo, "*Estudo Comparativo das API's JAX-RPC e FAXM na construção de Web Services*", VII Encontro de Estudantes de Informática, Palmas, 2005

[Santos 1996]
José P.O.Santos, João J. Ferreira, José M. Mendonça, "*Integrating Infrastructures for Manufacturing a Comparative Analysis*", in 6th international Conference on Integrated and Sustainable Industrial Productions Oe / ifIP / IEEE, Lisbon, Portugal, 1996

[Serrão 2004]
Carlos Serrão, Joaquim Marques, "*Programação com PHP 4.3*", FCA, 2004

[Silva 2003]
Pedro Miguel Abreu Silva, "Dissertação Mestrado Identificação e Controlo de Processos via Internet", Universidade Técnica de Lisboa, 2003

[SOAP 2003]
Simple Object Access Protocol version 1.2, 2003
Web site: http://www.w3.org/TR/soap/

[SOAP 2005a]
W3 Schools – W3C SOAP Activities, 2006
Web site: http://www.w3schools.com/w3c/w3c_soap.asp

[SOAP 2005b]
XML.com – A Brief Story of SOAP, 2005
Web site: http://webservices.xml.com/lpt/a/ws/2001/04/04/soap.html

[SPI 2000]
SPI – Sociedade Portuguesa de Inovação
Comércio Electrónico – Conceitos e Aplicações
Web site: http://spi.pt

[Tanenbaum 1992]
Tanenbaum Andrew S., "Modern Operating Systems", Prentice Hall International Editions, 1992

[UDDI 2006]
OASIS UDDI – Advancing Web Services Discovery Standards, 2006
Web site: http://www.uddi.org

[UDDI 2004]
UDDI Version 3.0.2 – UDDI Spec Technical Committee Draft, 2004
Web site: http://uddi.org/pubs/uddi_v3.htm

[UML 2006]
Unified Modelling Language, 2006
Web site: http://www.uml.org

[Vinoski 1997]
Steve Vinoski, "CORBA: Integrating Diverse Applications Within Distributed Heterogeneus Environments", IEEE Communication Magazine, Vol14, Nº.2, 1997

[XML 2004a]
Extensible Markup Language (XML) 1.0 (Third Edition), 2004
Web site: http://www.w3.org/TR/REC-xml

[XML 2004b]
Extensible Markup Language (XML) 1.1, 2004
Web site: http://www.w3.org/TR/xml11

[XML 2006]
Extensible Markup Language, 2006
Web site: http://www.w3.org/XML

[XML Schema 2004a]
XML Schema Part 0: Primer Second Edition, 2004
Web site: http://www.w3.org/TR/xmlschema-0/

[XML Schema 2004b]
XML Schema Part 1: Structures Second Edition, 2004
Web site: http://www.w3.org/TR/xmlschema-1/

[XML Schema 2004c]
XML Schema Part 2: Data types Second Edition, 2004
Web site: http://www.w3.org/TR/xmlschema-2/

[X500 2006]
The X.500 Directory – The mother of all directories
Web site: http://home20.inet.tele.dk/era/x500/x500-technology.htm

[W3C 2006]
World Wide Web Consortium, 2006
Web site: http://www.w3.org

[Wikipedia 2006]
Wikipédia – A enciclopédia livre
Web site: http://pt.wikipedia.org

[WSDL 2006]
WSDL Tutorial, 2006
Web site: http://www.w3schools.com/wsdl/default.asp

[WSDL 2006a]
Web Services Description Language (WSDL) Version 2.0 Part 0: Primer, 2006
Web site: http://www.w3.org/TR/wsdl20-primer

[WSDL 2006b]
Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, 2006
Web site: http://www.w3.org/TR/wsdl20

[WSDL 2004c]
Web Services Description Language (WSDL) Version 2.0 Part 2: Adjuncts, 2006
Web site: http://www.w3.org/TR/wsdl20-adjuncts

Índice de Apêndices

Apêndice A - Unified Modelling Language (UML) 123
A.1 Introdução 123
A.2 Cronologia UML 123
A.3 Elementos Visuais 125
A.3.1 Abstracções 125
A.3.2 Relacionamentos 127
A.3.3 Diagramas 130
A.3.4 Elementos Auxiliares 135
Apêndice B - Programação Peça ISO 137
B.1 Introdução 137
B.2 O Aparecimento do Comando Numérico 138
B.3 Uma Máquina com Comando Numérico 138
B.4 Conceitos de Programação CNC 142
B.5 Listagem das Funções Preparatórias e Funções M 145
B.6 Exemplo 149
Apêndice C - Mensagens Propostas 151
C.1 Mensagem HTML do Cliente para o Broker “Registo Cliente” 152
C.2 Mensagem HTML do Cliente para o Broker “Pedido_Orçamento” 153
C.3 Mensagem SOAP do Broker para o Servidor UDDI “find_tModelKey” 154
C.4 Mensagem SOAP do Servidor UDDI para o Broker “Devolve_tModelKey” 155
C.5 Mensagem SOAP do Broker para o Servidor UDDI “get_tModelKeyDetail (tModeKey_ProdSim82)” 156
C.6 Mensagem SOAP do Servidor UDDI para o Broker “Devolve_tModelKeyDetail(Loc_WSDL_ProdSim82)” 157
C.7 Mensagem SOAP do Broker para o Servidor UDDI “get_tModelKeyDetail (tModeKey_ProdSim83)” 158
C.8 Mensagem SOAP do Servidor UDDI para o Broker “Devolve_tModelKeyDetail(Loc_WSDL_ProdSim83)” 159
C.9 Mensagem SOAP do Broker para o Servidor UDDI “get_tModelKeyDetail (tModeKey_ProdSim84)” 160
C.10 Mensagem SOAP do Servidor UDDI para o Broker “Devolve_tModelKeyDetail(Loc_WSDL_ProdSim84)” 161
C.11 Mensagem SOAP do Broker para o Servidor UDDI “get_tModelKeyDetail (tModeKey_ProdSim85)” 162
C.12 Mensagem SOAP do Servidor UDDI para o Broker “Devolve_tModelKeyDetail(Loc_WSDL_ProdSim85)” 163

C.13 Mensagem SOAP do Broker para o Servidor UDDI “get_tModelKeyDetail (tModeKey_ProdSim86)” 164
C.14 Mensagem SOAP do Servidor UDDI para o Broker “Devolve_tModelKeyDetail(Loc_WSDL_ProdSim86)” 165
C.15 Mensagem SOAP do Broker para o Servidor UDDI “get_tModelKeyDetail (tModeKey_ProdReal_UA)” 166
C.16 Mensagem SOAP do Servidor UDDI para o Broker “Devolve_tModelKeyDetail(Loc_WSDL_ProdReal_UA)” 167
C.17 Mensagem do Broker para o Produtor Simulado 82 “Solicita Serviços WEB (WSDL) Produtor 82” 168
C.18 Mensagem SOAP do Produtor Simulado 82 para o Broker “Devolve Serviços WEB (WSDL) Produtor 82” 169
C.19 Mensagem SOAP do Broker para o Produtor Simulado 82 “Pedido_Orçamento_Produtor 82” 172
C.20 Mensagem SOAP do Produtor Simulado 82 para o Broker “Resposta_Orçamento_Produtor 82” 173
C.21 Mensagem do Broker para o Produtor Simulado 83 “Solicita Serviços WEB (WSDL) Produtor 83” 176
C.22 Mensagem SOAP do Produtor Simulado 83 para o Broker “Devolve Serviços WEB (WSDL) Produtor 83” 177
C.23 Mensagem SOAP do Broker para o Produtor Simulado 83 “Pedido_Orçamento_Produtor 83” 180
C.24 Mensagem SOAP do Produtor Simulado 83 para o Broker “Resposta_Orçamento_Produtor 83” 181
C.25 Mensagem do Broker para o Produtor Simulado 84 “Solicita Serviços WEB (WSDL) Produtor 84” 184
C.26 Mensagem SOAP do Produtor Simulado 84 para o Broker “Devolve Serviços WEB (WSDL) Produtor 84” 185
C.27 Mensagem do Broker para o Produtor Simulado 84 “Pedido_Orçamento_Produtor 84” 188
C.28 Mensagem SOAP do Produtor Simulado 84 para o Broker “Resposta_Orçamento_Produtor 84” 189
C.29 Mensagem do Broker para o Produtor Simulado 85 “Solicita Serviços WEB (WSDL) Produtor 85” 192
C.30 Mensagem SOAP do Produtor Simulado 85 para o Broker “Devolve Serviços WEB (WSDL) Produtor 85” 193
C.31 Mensagem SOAP do Produtor Simulado 85 para o Broker “Pedido_Orçamento_Produtor 85” 196
C.32 Mensagem do Produtor Simulado 85 para o Broker “Resposta_Orçamento_Produtor 85” 197

C.33 Mensagem do Broker para o Produtor Simulado 86 “Solicita Serviços WEB (WSDL) Produtor 86” 200
C.34 Mensagem SOAP do Produtor Simulado 86 para o Broker “Devolve Serviços WEB (WSDL) Produtor 86” 201
C.35 Mensagem SOAP do Broker para o Produtor Simulado 86 “Pedido_Orçamento_Produtor 86” 204
C.36 Mensagem SOAP do Produtor Simulado 86 para o Broker “Pedido_Orçamento_Produtor 86” 205
C.37 Mensagem do Broker para o Produtor Real U.A. Heidenhein “Solicita Serviços WEB (WSDL) Produtor UA” 208
C.38 Mensagem do Produtor Real U.A. Heidenhein para o Broker “Devolve Serviços WEB (WSDL) Produtor UA” 209
C.39 Mensagem SOAP do Broker para o Produtor Real U.A. Heidenhein “Pedido_Orçamento_Produtor UA” 212
C.40 Mensagem do Produtor Real U.A. Heidenhein para o Broker “Resposta_Orçamento_Produtor UA” 213
C.41 Mensagem HTML do Cliente para o Broker “Pedido_Encomenda” 216
C.42 Mensagem SOAP do Broker para o Produtor “Cria_Encomenda” 217
C.43 Mensagem SOAP do Produtor para o Broker “Confirmação_encomenda” 218
C.44 Mensagem HTML do Cliente para o Broker “Pedido Seguimento Encomenda” 222
C.45 Mensagem HTML do Cliente para o Broker “Pedido Seguimento Encomenda Específica” 223
C.46 Mensagem SOAP do Broker para o Produtor “Seguimento_Enc” 224
C.47 Mensagem SOAP do Produtor para o Broker “Estado_Enc” 225

Apêndice A - UNIFIED MODELLING LANGUAGE (UML)

A.1 Introdução

Neste apêndice pretende-se fazer uma abordagem simples e prática de como utilizar o UML na caracterização e modelação de sistemas, dado que foi utilizada nesta dissertação. A necessidade de utilização deste tipo de linguagem está directamente ligada à indispensável necessidade de descrever os sistemas informáticos complexos, a sua documentação e facilidade de leitura pelas várias entidades envolvidas tendo em vista a manutenção e evolução futura destes sistemas informáticos. As aplicações informáticas usadas por estas organizações também têm que ser flexíveis ao ponto de satisfazer as necessidades de informação dos seus potenciais utilizadores. Daí ter surgido a necessidade de utilizar uma linguagem genérica de modelação, ou seja, que permita o desenho ou esquematização de quaisquer tipos de modelos, independentemente da sua complexidade ou estrutura. Assim, a utilização da modelação ajuda o analista sistemas a compreender as funcionalidades do sistema, onde cada modelo, que é uma descrição do sistema cujos requisitos estão a ser analisados, têm também como objectivo facilitar e simplificar a comunicação com os clientes. A utilização de modelos ajudam na visualização do sistema como ele é ou como se pretende que venha a ser, permite especificar a estrutura ou o comportamento do sistema e documenta as decisões tomadas pelo integrador.

O UML é uma linguagem padrão que permite especificar, visualizar, construir e documentar todo e qualquer sistema de informação orientado por objectos. Pela abrangência e simplicidade dos conceitos utilizados, o UML simplifica a integração dos aspectos organizacionais que constituem um negócio e os elementos de natureza tecnológica que constituem o respectivo sistema informático. Assim, toda a complexidade do processo de dominação de regras de negócio, definição dos processos e fluxos de informação vêem-se apresentados de uma forma mais simples e clara para o integrador de sistemas informáticos. [UML 2006]

A.2 Cronologia UML

Os estudos sobre a tecnologia de objectos iniciaram-se na década de 1980 com ênfase nas linguagens de programação. No final da mesma década começaram a surgir os métodos de análise e projecto.

Os principais métodos foram de:

SHLAER & MELLOR (1989 e 1991);
COAD & YOURDON (1991);
COAD & NICOLA (1993);
COAD et al. (1995);
WIRFS-BROCK et al. (1990);
BOOCH (1994 e 1995);
RUMBAUGH (1991 e 1996);
MARTIN & ODELL (1994 e 1995);
JACOBSON (1994 e 1995).

Todos os métodos eram muito similares, apesar da existência de diferentes notações para representar o mesmo conceito, o que na verdade, causava muita confusão entre os técnicos e competição entre os metodologistas, o que provocou a "guerra dos métodos". Em 1994 James Rumbaugh e Grady Booch iniciaram o trabalho de unificação dos métodos Booch e OMT. Em 1995 no OOPSLA, Booch e Rumbaught apresentaram a documentação da versão 0.8 do método unificado e anunciaram a integração de Ivar Jacobson na equipa, formando assim o que eles denominaram de "Three Amigos". Durante 1996 eles trabalharam no método que passou a chamar-se Unified Modeling Language (UML) e o Object Management Group (OMG) iniciou um esforço para padronização na área de métodos. Dos esforços de Rumbaugh, Booch e Jacobson resultaram as versões 0.9 e 0.91 da UML em Junho e Outubro de 1996, respectivamente. A UML foi a sucessora da linguagem de modelagem encontrada nos métodos Booch, OOSE/Jacobson, OMT e outros como Modelos de Entidades e Relacionamentos. Cada um desses métodos é caracterizado por características bem definidas. O OOSE é orientado a '*Use Cases'* e providencia um excelente recurso para a análise de requisitos. A OMT foi pensada para análise e sistemas centrados a dados. Booch'93 foi relevante na fase de projecto e construção de sistemas voltados para Engenharia. Durante o ano de 1996, a Rational Software Corporation contou com a participação de outras instituições parceiras como: Hewlett-Packard, IBM, Microsoft, Oracle, Unisys, Platinum Technology, etc. Essa colaboração contribuiu para a produção da versão 1.0, uma versão bem definida, expressiva, poderosa e aplicável, a qual foi submetida a OMG para adopção. Em Setembro de 1997 liberaram a versão1.1 da UML e em Dezembro a mesma foi aprovada como padrão pela OMG.
Com a unificação dos métodos a UML alcançou dois objectivos:

- Acabou com as diferenças existentes entre os métodos anteriores;
- Unificou as perspectivas entre os diferentes tipos de sistemas, fases de desenvolvimento e conceitos internos.

A.3 Elementos Visuais

O UML recorre a uma notação padrão, constituída por um conjunto limitado de elementos de modelação de elementos de modelação, que podem ser descritos por diagramas, abstracções e relacionamentos. Os *diagramas* representam aspectos complementares de um sistema de informação e em cada um destes diagramas são utilizados símbolos que representam os elementos que estão a ser modelados, as chamadas *abstracções,* sendo os *relacionamentos* entre elas realizados por linhas. Os símbolos e linhas utilizadas têm um significado bem específico e possuem formas distintas, constituindo desta forma um modo de notação.

A.3.1 Abstracções

As *Abstracções* podem ser descritas como *Estruturais*, *Comportamentais*, de *Agrupamento* ou *Anotacionais*. As *Abstracções Estruturais* e *Comportamentais* reflectem a orientação por objectos da UML permitindo descrever a estrutura e o comportamento dos diversos elementos que constituem o sistema de informação. As *Abstracções de Agrupamento ou Anotacionais* são apenas conceptuais, podendo ser utilizadas para agrupar outros elementos estruturais, comportamentais ou mesmo de agrupamentos. Na Figura A-1 podem ser vistos alguns exemplos destas abstracções, as quais serão descritas individualmente.

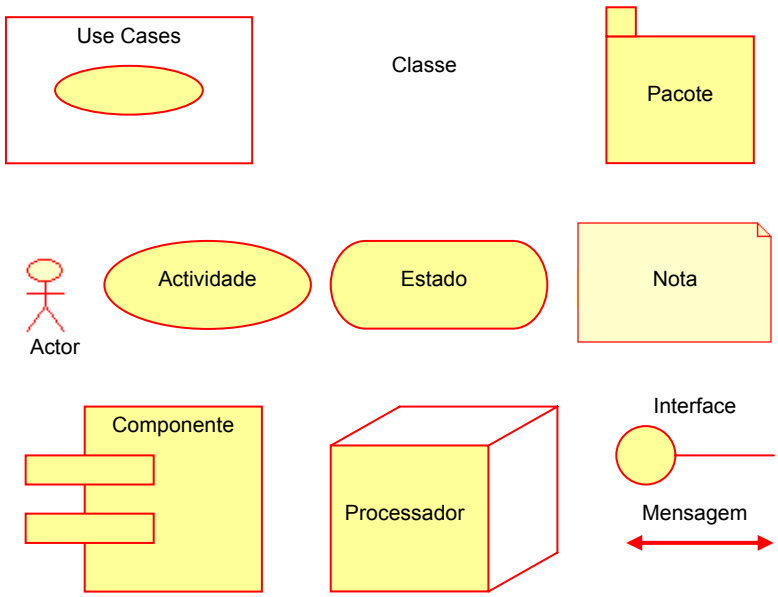


Figura A-1 – Exemplos de *Abstracções*

Use Cases *(casos de utilização)* – constituem a técnica em UML para representar o levantamento de requisitos de um sistema.

Classes – representam uma *Abstracção* sobre um conjunto de objectos que partilham a mesma estrutura e comportamento. Cada *Classe é* representada por um rectângulo e pode ser decomposta duas partes: a que contém os atributos e a que contém as funções (operações).

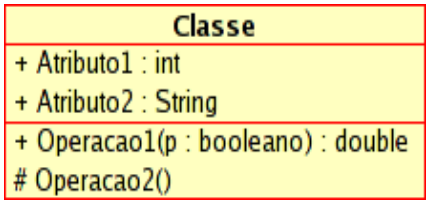


Figura A-2 – Exemplo de utilização de uma classe

Atributos das classes

No UML, os atributos são mostrados pelo seu nome, podendo também ser mostrados pelo tipo, valor inicial e outras propriedades. Os *Atributos* podem também ser exibidos pela sua visibilidade, utilizando a seguinte simbologia

+ indicam atributos públicos
# indicam atributos protegidos
- indicam atributos privados

Operações das classes

As *Operações* (métodos ou funções) são também são exibidos pelo seu nome podendo também ser representados pelos seus parâmetros e valores de retorno. As *Operações* podem, tal como os *Atributos*, mostrar sua visibilidade:

+ indicam operações públicas
# indicam operações protegidas
- indicam operações privadas

As *Classes* podem ter *Modelos*, ou seja, um valor que é usado para uma *classe* ou tipo não especificado. O tipo de *Modelo* é especificado quando uma classe é iniciada, isto é, quando um objecto é criado. Um exemplo de *Modelos* pode encontrar-se na programação em C++ moderno e Java 1.5 onde eles são chamados de *Genéricos*.

Pacotes – permitem dividir a complexidade de um sistema em partes mais pequenas para uma melhor gestão.

Actores – representam as entidades externas que interagem com o sistema.

Actividade – identifica o início do diagrama ou faz corresponder uma actividade operacional do sistema.

Estados – descrevem as operações que são executadas nesse estado.

Notas – permitem a inserção de comentários nos diagramas do sistema.

Componentes – são unidades modulares, com interfaces bem definidas, que podem ser substituídas dentro do seu ambiente. As suas dependências em relação a outros elementos são definidas de modo a que possam ser tratados o mais independentemente possível, podendo ser reutilizados e substituídos através das interfaces que requerem e disponibilizam.

Processador – representa, de uma forma geral, os blocos das entidades onde são executadas operações ou acções.

Interfaces – representam o conjunto de serviços disponibilizados pelo componente, adaptados às necessidades e formatos particulares do subsistema.

Mensagens – representam a invocação de um serviço (operação) disponibilizado por um objecto, com o objectivo de realizar uma acção ou operação.

A.3.2 Relacionamentos

Os objectos têm relações entre eles: um professor lecciona uma disciplina para alunos numa sala, um cliente faz uma reserva de alguns lugares para uma data, um cliente tem uma ou mais encomendas em execução ou em espera, etc. A modelação UML reconhece cinco tipos de relacionamentos: a *Associação*, a *Generalização* (ou herança), a *Dependência*, a *Agregação* e a *Composição*.

Estes objectos de relacionamento apresentam-se sob a forma gráfica como se pode ver na Figura A-3:

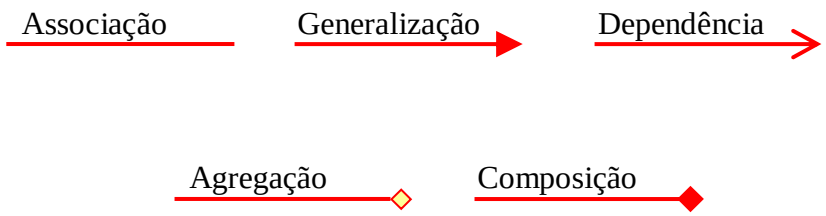


Figura A-3 – Exemplo da representação de relacionamentos

A.3.2.1 Associação

Uma *Associação* representa um relacionamento entre *Classes* e fornece a semântica comum e a estrutura para muitos tipos de "ligações" entre objectos. *Associações* são o mecanismo que permite aos objectos comunicarem entre si. Elas descrevem a ligação entre diferentes *Classes* (a ligação entre os objectos actuais é chamada ligação do objecto ou *link*). As *Associações* podem ter uma regra que especifica o propósito da associação e pode ser uni ou bidirecional (indicando se os dois objectos participantes do relacionamento podem mandar mensagens um para o outro, ou se apenas um deles tema a capacidade de enviar ou receber mensagens). Em UML, as *Associações* são representadas como linhas unindo as *Classes* participantes do relacionamento e podem mostrar a regra e a multiplicidade de cada um dos participantes, que dita como os objectos de um dos lados da associação se relaciona com o outro. A multiplicidade é exibida como um intervalo [min...max] de valores não negativos com uma estrela (*) no lado máximo representando infinito.

Figura A-4 – Exemplo da representação de um relacionamento de associação

A.3.2.2 Generalização ou Herança

A *Generalização* ou *Herança* é um dos conceitos fundamentais da programação orientada ao *Objeto*, na qual uma classe "herda" todos os atributos e operações da classe da qual deriva e pode escrever e/ou modificar alguns deles, bem como adicionar mais atributos e operações próprias. Em UML, uma associação *Generalização* entre duas classes coloca-as numa hierarquia, representando o conceito de herança de uma classe derivada de uma classe base.

Como se pode ver na Figura A-5, as *Generalizações* são representadas por uma linha ligando duas classes, com uma seta no lado da classe base.

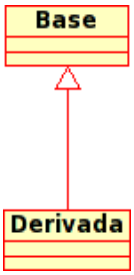


Figura A-5 – Exemplo da representação de um relacionamento de generalização

A.3.2.3 Dependência

Os relacionamentos de *Dependência* são relacionamentos de utilização no qual uma mudança na especificação de um elemento pode alterar a especificação do elemento dependente. A dependência entre classes indica que os objectos de uma classe usam serviços dos objectos de outra classe. (Figura A-6

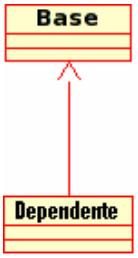


Figura A-6 – Exemplo da representação de um relacionamento de dependência

A.3.2.4 Agregação

As *Agregações* são um tipo especial de associação no qual as duas classes participantes não possuem um nível igual, mas fazem um relacionamento “todo-parte”. Uma *Agregação* pode ser descrita como uma classe que possui a regra do todo e é composta por outras classes que possuem a regra das “partes”. Para as Agregações, a classe que age como o todo tem sempre uma multiplicidade de um. Em UML, *Agregações* são representadas por uma associação que mostra um losângulo no lado do todo. (Figura A-7)

Figura A-7 – Exemplo da representação de um relacionamento de agragação

A.3.2.5 Composição

As *Composições* são associações que representam *Agregações* muito fortes, isto é, formam relacionamentos todo-parte, mas o relacionamento é tão forte que as partes não podem existir independentes. Elas existem somente dentro do todo e se o todo é destruído as partes são também destruídas. Em UML, as *Composições* são representadas por um losângulo sólido no lado do todo, como se pode ver na Figura A-8.

Figura A-8 – Exemplo da representação de um relacionamento de composição

A.3.3 Diagramas

Os elementos UML são usados para criar diagramas que representam uma determinada parte ou um ponto de vista do sistema. Estes diagramas podem descrever um sistema de uma forma completa ou parcial. Consoante o tipo de sistema, a informação que se pretende analisar e os fluxos de informação podem ser utilizados vários tipos de diagramas podendo ser disponibilizados da seguinte forma:

Diagramas de Casos de Uso – são diagramas que identificam as fronteiras do sistema (os cenários do sistema) descrevendo os serviços de cada um dos diversos utilizadores (pessoas ou outros utilizadores do sistema). Estes diagramas servem principalmente para simplificar a comunicação entre os futuros utilizadores do sistema, sendo especialmente úteis para determinar os recursos necessários que o sistema deve ter. De uma forma concreta, indicam o que o sistema deve fazer mas não faz, não especificando como isso é conseguido. Um exemplo deste tipo de diagramas encontra-se representado na Figura A-9.

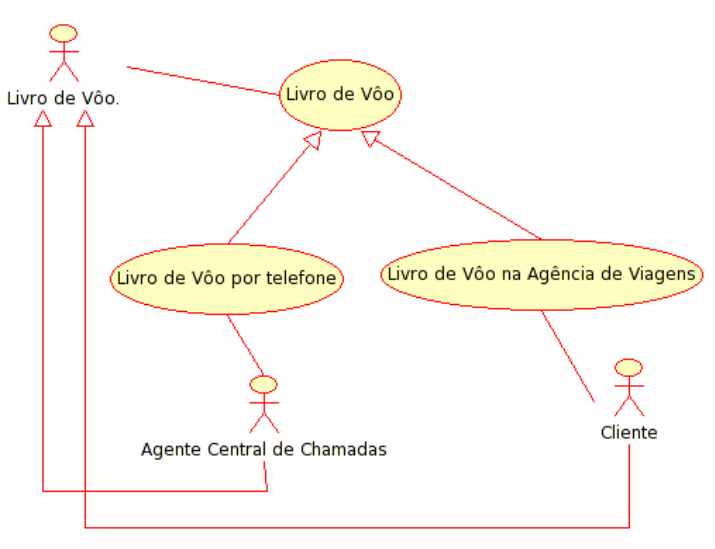


Figura A-9 – Exemplo do Diagrama de Casos de Uso

Diagramas de Classes – são diagramas que descrevem a estrutura da informação utilizada no sistema. Mostram as diferentes classes que constituem um sistema e como elas se relacionam. Estes diagramas são também chamados diagramas estáticos porque mostram as classes com as suas funções e atributos, bem como os relacionamentos estáticos que estabelecem, isto é, que classes "conhecem" outras classes ou que classes "são parte" de outras classes, mas não mostram a troca de mensagens ou informações entre elas. A Figura A-10 exemplifica um diagrama de classes.

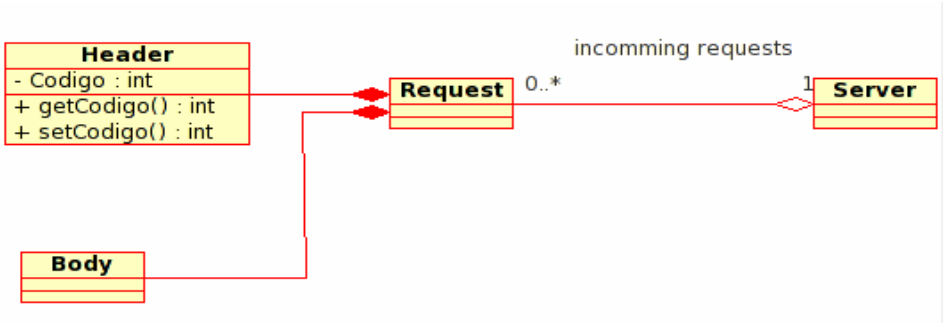


Figura A-10 – Exemplo do Diagrama de Classes

Diagrama de Objectos – são diagramas através dos quais podem ser ilustrados diagramas de classes concretos.

Diagrama de Sequência – são diagramas que ilustram como os objectos do sistema interagem, para no final poderem fornecer a funcionalidade do sistema. Os *Diagramas de Sequência* mostram a troca de mensagens (*Métodos*) entre os diversos *Objectos*, numa

situação específica e delimitada no tempo. Os Objectos são instâncias de classes. Os *Diagramas de Sequência*, são também normalmente chamados Diagramas de Interacção e colocam uma ênfase especial na ordem e nos momentos em que as mensagens são enviadas para os *Objectos*. Nos *Diagramas de Sequência,* os *Objectos* são representados através de linhas verticais tracejadas, com o seu nome no topo. O eixo do tempo é também vertical, aumentando para baixo, de modo que as mensagens são enviadas de um *Objecto* para outro, representando-se através de setas com a operação e o nome dos parâmetros. Um exemplo deste tipo de diagramas encontra-se ilustrado na Figura A-11.

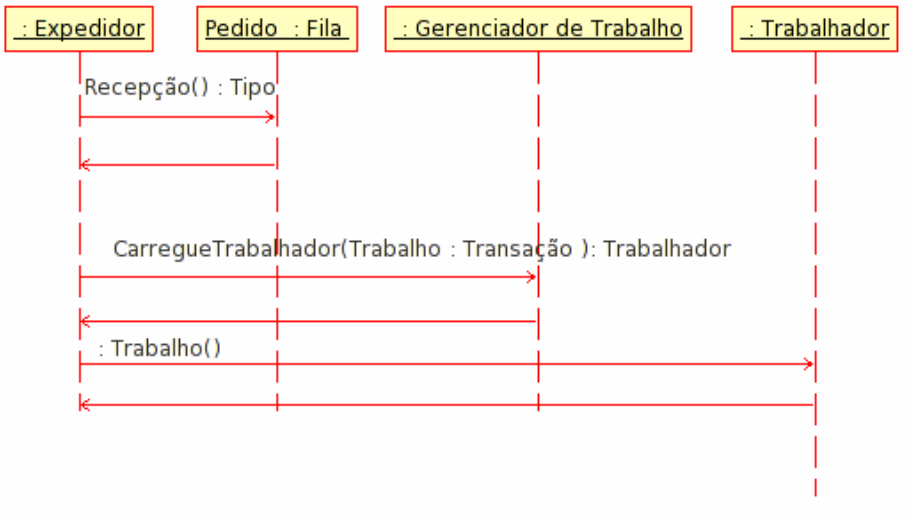


Figura A-11 – Exemplo de um Diagrama de Sequência

Diagrama de Colaboração – estes diagramas mostram as interacções que ocorrem entre os objectos participantes numa situação específica, ou seja, é mais ou menos a mesma informação mostrada pelos *Diagramas de Sequência.* Enquanto que nos os *Diagramas de Colaboração* são destacados os relacionamentos entre os objectos e a sua topologia, nos *Diagramas de Sequência* a ênfase é colocada no modo de como as interações ocorrem no tempo. Nos *Diagramas de Colaboração,* as mensagens enviadas de um objeto para outro são representadas por setas, mostrando o nome da mensagem, parâmetros e a sequência da mensagem. Estes diagramas são especialmente indicados para mostrar um fluxo ou situação específica do programa e são um dos melhores tipos de diagrama para rapidamente demonstrar ou explicar um processo na lógica do programa. Um exemplo de Diagramas de colaboração encontra-se representado na Figura A-12.

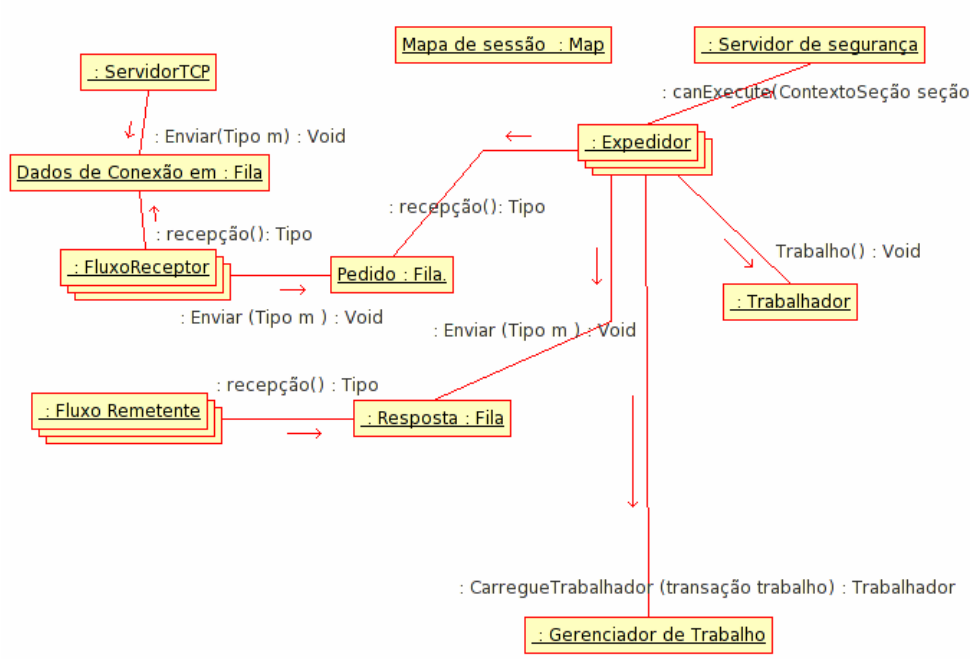


Figura A-12 – Exemplo de um Diagrama de Colaboração

Diagramas de Actividade – estes diagramas descrevem cada um dos *Use Cases*, realçando o encadeamento de actividades realizadas por cada um dos *Objectos* do sistema, na óptica do fluxo de trabalho. Os *Diagramas de Actividade* são similares aos *Diagramas de Fluxo de Procedimentos* diferenciando-se pelo facto de todas as *Actividades* serem claramente anexas aos *Objectos.* Além disso, estão sempre associados a uma *Classe*, a uma *Operação* ou a um *Caso de Uso*. Os *Diagramas de Actividade* suportam *Actividades* sequenciais assim como paralelas. A execução paralela é representada pelos ícones Forquilha/Esperar e para as *Actividades* executadas em paralelo não é importante a ordem pela qual elas se executam (podem ser executadas ao mesmo tempo ou uma após a outra). *E*ntende-se por Actividade um passo simples num processo, um estado de sistema com atividade interna com, pelo menos, uma transição de saída. As *Actividades* também podem ter mais que uma transição de saída se possuirem condições diferentes. Podem ser formadas hierarquias, isto é, uma Actividade pode ser composta por diversas *Actividades* em "detalhe", na qual as transições de entrada e saída devem corresponder às transições de entrada e saída do diagrama de detalhe. Na Figura A-13 encontra-se um exemplo dos *Diagramas de Actividade*

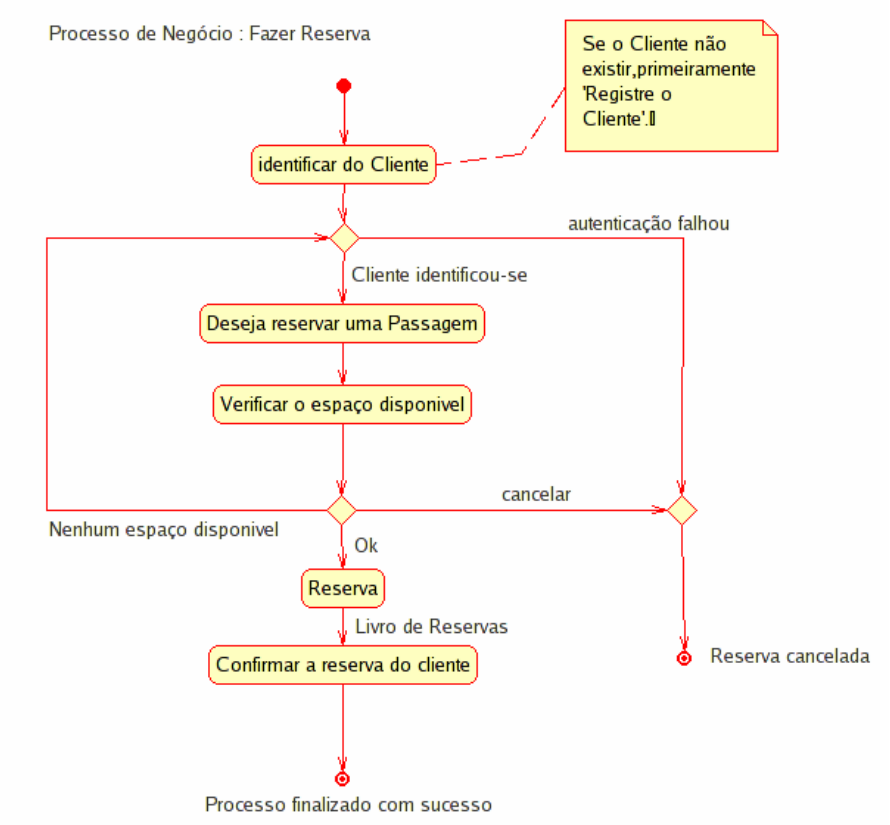


Figura A-13 – Exemplo de um Diagrama de Actividade

Diagramas de Estados – estes diagramas permitem modelar o comportamento dos objectos, isto é, servem para descrever alterações nos valores dos atributos dos objectos em resultado da ocorrência de certos eventos (Figura A-14). Por exemplo um tipo de *Objecto Servidor/Rede* pode estar num dos seguintes estados durante a sua vida:

- Pronto
- Escutar
- Processar
- Parado

e os eventos que podem fazer com que o *Objecto* mude de estado são:

- Criação do Objecto
- O *Objecto* recebe mensagem ouvir
- Um cliente solicita uma ligação através da rede
- Um cliente termina um pedido
- O pedido é executado e terminado
- O *Objecto* recebe a mensagem de parar

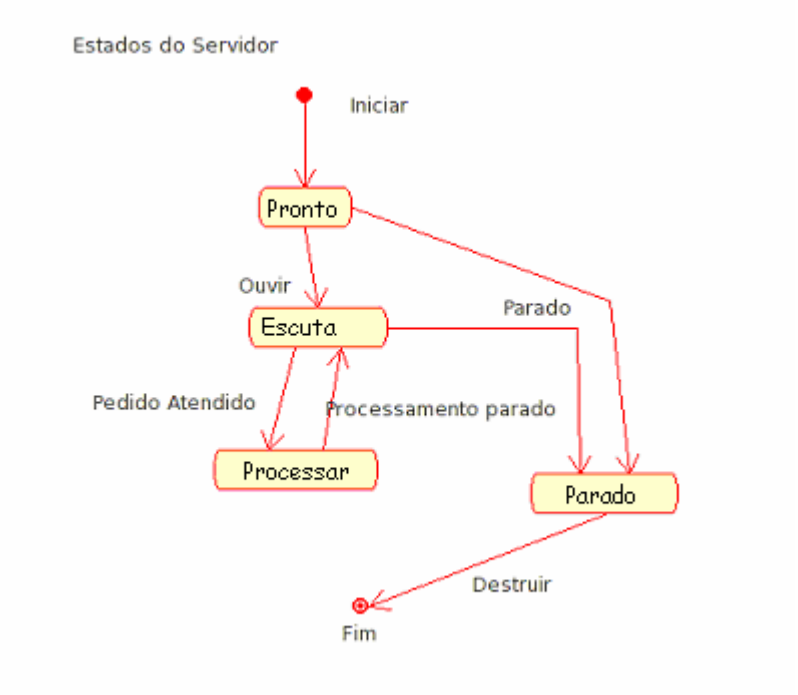


Figura A-14 – Exemplo de um Diagrama de Estado

Diagramas de Componentes – *estes diagramas descrevem* a arquitectura da aplicação informática em termos de componentes de software, isto é, mostram os componentes de programação de alto nível e os artefactos de que eles são feitos como por exemplo os arquivos de código fonte, bibliotecas de programação ou tabelas de bancos de dados relacionais.

Diagramas de Instalação – Estes diagramas permitem descrever a arquitectura dos equipamentos informáticos utilizados e atribuem os componentes da aplicação aos diversos equipamentos.

A.3.4 Elementos Auxiliares

Existem dois elementos em UML que não possuem nenhum valor real semântico para o modelo, mas ajudam a elucidar partes do diagrama. Estes elementos são:

- Linhas de texto – permitem adicionar informações de texto nos diagramas. São textos livres e não possuem nenhum significado para o *Modelo* propriamente dito.

- Notas de Texto e Âncoras – permitem adicionar informações mais detalhadas sobre um *Objecto* ou situação específica. Possuem a grande vantagem de poderem ser

ancoradas a elementos UML para mostrar que a nota “pertence” a um *Objecto* ou a uma situação específica.

- Caixas - são retângulos de forma livre que podem ser usados para agrupar itens, tornando os diagramas mais legíveis. Não possuem nenhum significado lógico no *Modelo*.

Apêndice B - PROGRAMAÇÃO PEÇA ISO

B.1 Introdução

Neste apêndice faz-se uma breve abordagem às máquinas CNC com comando numérico e sua programação porque foi necessário, no âmbito deste trabalho de dissertação, desenvolver um programa peça de forma a permitir maquinar uma peça, quando é realizado um pedido de encomenda no *Produtor Real*. (centro de maquinação Heidenhein 426PB)

As máquinas ferramentas têm desempenhado um papel fundamental, em diversas áreas, no desenvolvimento tecnológico do mundo. Muitas das maquinarias, ainda que projectadas e concebidas, não tiveram a evolução e expansão desejadas por não existirem, no passado, meios tecnológicos capazes de as fabricarem a um nível industrial. Caso particular como o da máquina a vapor, inventada por James Watt em 1766, só teve o desenvolvimento industrial em grande escala passados 10 anos, quando em 1976, John Wilkinson construiu a primeira mandriladora. A partir desse momento passou a ser possível produzir uma ampla gama de produtos em grandes quantidades e a preços mais acessíveis.

O constante desenvolvimento das máquinas ferramentas sempre procurou soluções que permitissem aumentar a produtividade. Uma das soluções passou por agrupar no mesmo local as máquinas ferramentas do mesmo tipo ou que fizessem a mesma operação, sendo cada uma delas operada independentemente. Com o aumento de complexidade das peças, surgiu a necessidade utilizar um maior número e tipo de operações de maquinação, sendo cada uma delas realizada na máquina mais adequada. A grande variedade de peças solicitadas diariamente assim como as técnicas utilizadas para as produzir, obrigava a uma constante reconfiguração das ferramentas. Assim, começou a pensar-se numa forma de reduzir o número de máquinas e evitar as constantes mudanças de configuração das máquinas. Com este propósito, surgiu a ideia de conceber uma máquina capaz de realizar as várias operações maquinação necessárias para a produção da peça, reduzindo assim o espaço ocupado pelas diferentes máquinas e mão-de-obra utilizada em cada uma delas.
A electrónica, hidráulica, pneumática, automação foram adoptadas como tecnologias para automatizar a máquina ferramenta capaz de produzir, por ela própria, peças complexas e de difícil fabricação para o ser humano associada ao baixo custo de produção necessário para garantir a competitividade no mercado. Nos dias que correm verifica-se uma constante evolução e utilização de novas tecnologias com o objectivo de optimizar factores como a precisão, produtividade, flexibilidade e rapidez de produção das máquinas ferramentas. De uma forma genérica, este passou a ser o conceito muito inicial de centro de maquinação, onde se pretendeu realizar numa só máquina as várias operações necessárias à produção de uma

peça. Todas essas operações e trocas de ferramenta passaram a ser controladas por comando numérico o que permitiu aumentar a cadência de produção, qualidade de acabamento e redução de tempos de "setup" (ex. troca de ferramentas). Hoje, pode-se afirmar veemente que a evolução tecnológica é o factor chave do crescimento e da competitividade das empresas.

B.2 O Aparecimento do Comando Numérico

O aparecimento do comando numérico data de 1947 e foi idealizado e criado por John Parsons em Michigan nos E.U.A.. A "Parsons Corporation" fabricava essencialmente equipamentos para a indústria da defesa e após várias tentativas de implementar diferentes métodos de fabrico, teve a ideia de experimentar a utilização de um computador para comandar uma máquina fresadora que se movia em pequenos espaços, seguindo uma trajectória previamente definida. A ideia suscitou um enorme interesse junto da Força Aérea dos E.U.A. que tinha a necessidade de fabricar componentes complexos, e que sofriam constantes alterações, em curtos espaços de tempo. Em 1949 foi estabelecido um acordo entre "U.S. Air Force", a "Parsons Corporation e o "Instituto Tecnológico de Massachusetts" (M.I.T.), sendo 3 anos mais tarde, em 1952, apresentada o primeiro protótipo de uma fresadora com comando numérico de 3 eixos. Posteriormente foram desenvolvidos numerosos tipos de comandos numéricos, cada vez mais sofisticados mas com diversos problemas, complexos, complicados, onerosos e de difícil programação. A evolução da informática, dos microcontroladores, da microelectrónica e da automação permitiu aumentar o rendimento e funcionalidade dos comandos numéricos.

B.3 Uma Máquina com Comando Numérico

Um comando numérico pode ser definido como sendo um equipamento capaz de receber informações através de um periférico próprio para entrada de dados, compilar estas informações e comandar as trajectórias de corte da máquina (mecânica) de modo a que esta realize as operações automaticamente numa sequência programada e sem intervenção do operador.

As máquinas com comando numérico apresentam várias vantagens das quais se destacam:

- Aumento da produtividade das máquinas – deve-se essencialmente à diminuição do tempo de maquinação devido à diminuição dos tempos de deslocamento em vazio e da rapidez dos posicionamentos controlados pelos sistemas electrónicos.

- Flexibilidade – permite realizar um maior número de operações que os sistemas convencionais. Uma simples mudança de programa pode ser suficiente para produzir uma peça completamente diferente.

- Precisão – permite realizar todas as operações necessárias à produção da peça na mesma máquina não havendo necessidade de recorrer a várias fixações em várias máquinas. A precisão passa a depender apenas de uma máquina e não das várias máquinas que convencionalmente teriam que ser usadas para realizar as operações necessárias à produção da peça. São providas de elementos mecânicos, eléctricos e electrónicos tecnologicamente mais avançados e recentes.

- Redução de Controlos – permite realizar a maquinação de uma peça seguindo sempre as mesmas trajectórias conseguindo-se uma qualidade de produção constante, dependendo esta do desgaste das ferramentas. Com isto é possível reduzir o número de controlos intermédios, nomeadamente os dimensionais.

- Viabilização – permite produzir peças cada vez mais complexas.

- Segurança – permitem realizar todas as operações necessárias para a produção de uma peça e estão providas de blindagens de resguardo e sistemas de segurança.

Uma máquina de comando numérico é normalmente constituída pelos elementos ilustrados na Figura B-1:

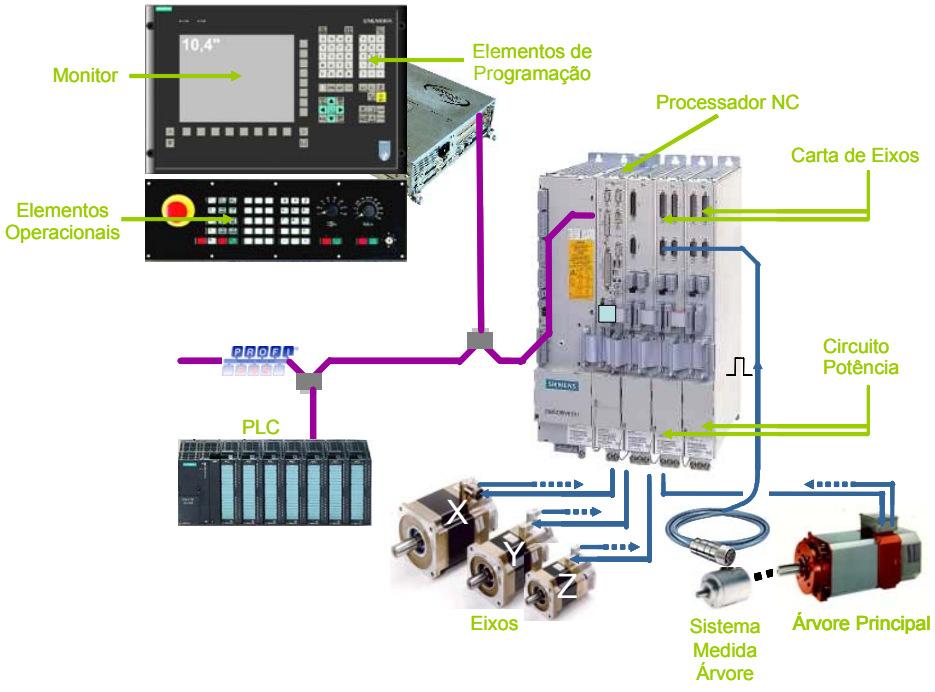


Figura B-1 – Elementos que constituem uma máquina controlada por CN

- Processador CN – onde são processados os dados existentes em memória e o programa CN que são posteriormente transferidos à máquina através de impulsos de comando.

- Monitor / Vídeo / Display – onde é visualizada a informação existente no comando.

- Elemento de Programação – Teclado alfanumérico e numérico onde se podem introduzir os comandos ou escrever programa de comando numérico.

- Elemento Operacionais – onde se encontram os comandos que permitem realizar operações simples (ligar/desligar máquina, deslocar eixos, controlar velocidades de avanço e rotação (override e handweel, etc.).

- PLC (Programable Logic Controler) – onde são analisadas as condições, tais como sinais provenientes dos sensores e actuadores, que permitirão controlar e monitorizar o funcionamento do CNC.

- Carta de Eixos – onde se faz a combinação entre o sistema de medição e o accionamento dos eixos.

- Circuito de Potência – onde os impulsos provenientes do processador e do controlo da interface são amplificados de forma a accionarem os eixos.

- Aparelho Externos – onde se podem incluir leitores de disquetes, gravadores de fitas magnéticas, sistemas de verificação de quebra de ferramentas, etc.

- Arvore Principal – elemento rotacional que conjuntamente com uma ferramenta é a responsável pelas operações a realizar na peça (maquinação, furação, desbaste, fresagem, etc.) (Figura B-2).

Figura B-2 – Exemplo da Árvore Principal

- Eixos de Avanço – elementos responsáveis por posicionar a peça ou a árvore principal nas posições das operações de maquinação. Normalmente são elementos mecânicos (fusos/porca de esferas) accionados e controlados por motores eléctricos e seus sistemas de medição (réguas ópticas ou encoders) (Figura B-3). Máquinas mais recentes já começam a utilizar motores lineares em substituição do sistema convencional de fusos.

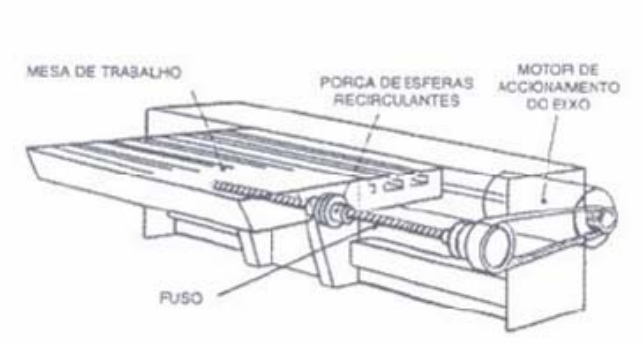


Figura B-3 – Exemplo da mecânica de um eixo

- Dispositivos de Fixação de Peças – onde se fixam as peças na mesa de trabalho. Têm papel importante na precisão e qualidade final da maquinação.

- Dispositivos de Troca de Ferramentas – armazém onde se encontram as ferramentas. Permite a troca, manual ou automática, das ferramentas a utilizar durante a maquinação de uma peça.(Figura B-4)

Figura B-4 – Exemplo de um armazém de ferramentas

B.4 Conceitos de Programação CNC

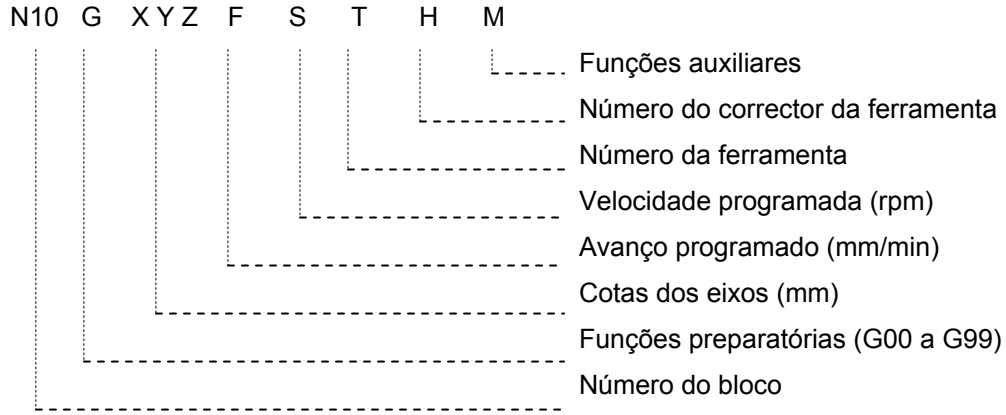
Com o aparecimento de máquinas CNC (Computer Numerical Control) a relação do homem/máquina alterou-se. A optimização e fiabilização das máquinas CNC aumentavam significativamente e o homem começou a ser confrontado com uma máquina de grande potencial da qual começou a deixar de ter controlo. Os utilizadores destas máquinas começaram a ser confrontados com novos problemas:

- Como conseguir tornar acessíveis estes avanços tecnológicos ao maior número de pessoas?

- Como conseguir que a programação necessária e sua implementação acompanhasse a evolução destas máquinas?

Assim, surgiu a necessidade de tornar acessível ao maior número de pessoas uma linguagem de programação, compreensiva e de fácil implementação que permita controlar as potencialidades destas máquinas possuidoras de comando numérico. Existem várias linguagens de programação sendo a mais utilizada a linguagem definida pela norma ISO DIN 66025. A linguagem ISO está especialmente indicada para controlar os movimentos dos eixos, já que proporciona informações e condições de movimento e avanço. Utilizando esta linguagem passou a ser possível escrever programas peça com as trajectórias, as ferramentas, as velocidades e os avanços de corte, etc.

A programação de máquinas CNC pode ser efectuada directamente no teclado da máquina ou através de outros periféricos tais como computadores pessoais ou em máquinas mais antigas: disquetes, leitor de fitas magnéticas, etc. Um programa peça de um comando numérico é constituído por um conjunto de blocos ou instruções, formados por caracteres numéricos e alfanuméricos, onde constam todos os dados geométricos e tecnológicos necessários para que uma máquina ferramenta execute as operações e movimentos desejados de uma forma sequencial e controlada. Em cada um destes blocos podem ser encontradas:



- Número da Sequência (N) – este termo corresponde ao número da sequência do bloco de programação e servem para identificar cada um dos blocos que constituem o programa ou para quando se realizam referências de saltos. Todos os blocos de um programa são numerados, representando-se com a letra N seguida de 4 números, podendo ser ou não seguidos. É normalmente utilizada uma numeração em intervalos de 10, deixando assim possibilidade para futuras alterações no programa.

- Funções Preparatórias (G) – estes termos servem para determinar a geometria e condições de trabalho (tipo de trajectória, interpolações lineares e circulares, unidades do sistema, ciclos automáticos, etc.) durante a operação de maquinação. As funções preparatórias iniciam-se sempre pela letra G seguida de 2 algarismos G[xx] e programam-se sempre no início do bloco. Podem ser do tipo Modais ou Não Modais. As primeiras são funções que uma vez programadas permanecem na memória do CN, ficando activas para os blocos seguintes, até serem modificadas por outra função do mesmo grupo. As segundas são funções que têm que ser programadas todas as vezes que são requeridas, isto é, apenas não válidas no bloco onde se encontram. As funções preparatórias encontram-se divididas em vários grupos, ordenadas pela sua função.

- Coordenadas dos Eixos (X Y Z) – estes termos indicam as coordenadas dos pontos finais do deslocamento de uma trajectória. As cotas dos eixos podem ser programadas em milímetros ou polegadas, em coordenadas absolutas ou incrementais, consoante as funções preparatórias G usadas. Podem ainda ser definidas como parâmetros, sendo os seus valores atribuídos na respectiva página de atribuição.

  X50 Y40 Z30 – atribuição numérica
  XP90 YP91 ZP92 – atribuição paramétrica (P90, P92, P92)

- Velocidade de Avanço (F) – este termo é utilizado para quantificar o valor da velocidade de avanço da ferramenta durante a maquinação. Depende do tipo de material, precisão pretendida, ferramentas utilizadas, entre outros factores. Dependendo da função preparatoria G utilizada, a sua declaração também varia. Se utilizar:

  G94 F1000 – a ferramenta vai movimentar-se com um avanço de 1000mm/min
  G95 F0,5 – a ferramenta vai movimentar-se com um avanço de 0,5mm/rot

  Os avanços programados podem variar de 0% a 120% mediante a variação do comutador dos avanços, existente no quadro de comandos da máquina, desde que as funções G33, G47 ou G84 não estejam activas.

- Velocidade de Rotação (S) – este termo determina a velocidade da rotação da árvore principal, podendo também, dependendo do tipo de máquina ferramenta, programar a velocidade de corte.

- Número de Ferramentas (T) – este termo indica o número da ferramenta seleccionada para executar a operação definida na sequência.

- Correctores de Ferramentas (H). – este termo é usado para designar o corrector de ferramenta a usar, onde se fixa o valor da compensação automática do raio ou comprimento da ferramenta.

- Vectores de Posição do Centro para a Interpolação Circular (I J K) – Este termo indica a coordenada do centro da interpolação circular, onde:

    - I – o vector de posição do centro do arco medido no eixo X.
    - J – o vector de posição do centro do arco medido no eixo Y.
    - K – o vector de posição do centro do arco medido no eixo Z.

- Funções Auxiliares (M)– estes termos utilizam-se para indicar à máquina outras funções complementares, tais como:

    - Paragens do programa
    - Sentido de rotação da árvore principal
    - Mudança de ferramenta
    - Ligar ou desligar determinadas funções ou acessórios

- Expressões de programação (IF, GOTO, CALL), expressões aritméticas (+, -, *, /) e lógicas e relacionais (<, >, =) que se assemelham às terminologias utilizadas por outras linguagens de programação.

B.5 Listagem das Funções Preparatórias e Funções M

	Instruções de Interpolação	
Nome	**Significado**	**Modal**
G00	Posicionamento rápido (até 5 eixos)	Sim
G01	Interpolação linear (até 5 eixos)	Sim
G02	Interpolação circular (sentido horário)	Sim
G03	Interpolação circular (sentido anti-horário)	Sim
G33	Roscagem	Sim

	Ciclos fixos	
Nome	**Significado**	**Modal**
G60	Ciclo fixo múltiplo em forma de linha recta	Não
G61	Ciclo fixo múltiplo em forma de paralelogramo	Não
G62	Ciclo fixo múltiplo em forma de malha	Não
G63	Ciclo fixo múltiplo em forma de círculo	Não
G64	Ciclo fixo múltiplo em forma de arco	Não
G65	Ciclo fixo múltiplo em forma de corda de arco	Não
G66	Ciclo fixo em caixas irregulares com ou sem ilhas	Não
G67	Ciclo fixo de desbaste de caixas irregulares	Não
G68	Ciclo fixo de acabamento de caixas irregulares	Não
G69	Ciclo fixo de furação profunda com passo variável	Sim
G79	Modificação dos parâmetros de um ciclo fixo	Não
G80	Anulação de qualquer ciclo fixo	Sim
G81	Ciclo fixo de furação simples	Sim
G82	Ciclo fixo de furação com temporização	Sim
G83	Ciclo fixo de furação profunda com passo constante	Sim
G84	Ciclo fixo de roscagem rígida com macho	Sim
G85	Ciclo fixo de escariado	Sim
G86	Ciclo fixo de mandrilado com retrocesso em G00	Sim
G87	Ciclo fixo de caixa rectangular ou quadrada	Sim
G88	Ciclo fixo de caixa circular	Sim
G89	Ciclo fixo de mandrilado com retrocesso em G01	Sim
G98	Retrocesso da ferramenta ao plano de partida	Sim
G99	Retrocesso da ferramenta ao plano de referência	Sim

	Condições de Mecanização	
Nome	**Significado**	**Modal**
G04	Paragem temporizada	Não
G05	Aresta morta	Sim
G07	Aresta viva	Sim
G50	Aresta morta controlada	Não
G51	Look.ahead (acelerador para blocos DNC)	Não
G70	Programação de cotas em polegadas	Sim
G71	Programação de cotas em milímetros	Sim
G90	Programação em cotas absolutas	Sim
G91	Programação em cotas incrementais	Sim
G92	Pré-selecção de cotas	Não
G94	Avanço em mm/minuto	Sim
G95	Avanço em mm/rotação	Sim
G96	Velocidade de avanço superficial constante	Sim
G97	Velocidade do centro da ferramenta constante	Sim

	Ajudas à programação	
Nome	**Significado**	**Modal**
G06	Centro da circunferência em coordenadas absolutas	Não
G08	Trajectória circular tangente à trajectória anterior	Não
G09	Circunferência definida por três pontos	Não
G10	Anulação da imagem de espelho	Sim
G11	Imagem espelho no eixo X	Sim
G12	Imagem espelho no eixo Y	Sim
G13	Imagem espelho no eixo Z	Sim
G14	Imagem espelho nas direcções programadas	Sim
G36	Arredondamento controlado das arestas	Não
G37	Entrada tangencial da ferramenta	Não
G38	Saída tangencial da ferramenta	Não
G39	Corte controlado de arestas	Não
G72	Factor escala	Sim
G73	Rotação do sistema de coordenadas	Sim
G77	Acoplamento electrónico dos eixos	Sim
G78	Anulação do acoplamento electrónico dos eixos	Sim

	Sistema de cópia	
Nome	**Significado**	**Modal**
G23	Activação da cópia	Não
G24	Activação da digitalização	Não
G25	Desactivação da cópia/digitalização	Não
G26	Calibragem da sonda de cópia	Não
G27	Definição do contorno de cópia	Não
G75	Movimento do apalpador até tocar	Não
G76	Movimento do apalpador até deixar de tocar	Não

	Compensação das dimensões da ferramenta	
Nome	**Significado**	**Modal**
G15	Selecção eixo perpendicular ao plano de trabalho	Sim
G40	Cancela a compensação do raio da ferramenta	Sim
G41	Compensação do raio da ferramenta (esquerda)	Sim
G42	Compensação do raio da ferramenta (direita)	Sim
G43	Compensação do comprimento da ferramenta	Sim
G44	Anulação da compensação comprimento da ferramenta	Sim

	Sistemas de coordenadas	
Nome	**Significado**	**Modal**
G16	Selecção do plano principal por 2 direcções	Sim
G17	Plano de trabalho X – Y	Sim
G18	Plano de trabalho X – Z	Sim
G19	Plano de trabalho Y – Z	Sim
G20	Definição limites inferiores da zona de trabalho	Não
G21	Definição limites superiores da zona de trabalho	Não
G22	Activa e desactiva as zonas de trabalho	Não
G53	Programação em relação ao zero máquina	Não
G54	Mudanças de origem absoluta 1 (zero peça)	Sim
G55	Mudanças de origem absoluta 2 (zero peça)	Sim
G56	Mudanças de origem absoluta 3 (zero peça)	Sim
G57	Mudanças de origem absoluta 4 (zero peça)	Sim
G58	Mudanças de origem incremental 1 (zero peça)	Sim
G59	Mudanças de origem incremental 2 (zero peça)	Sim
G74	Busca automática de referência máquina	Não
G92	Pré selecção de cotas	Não
G93	Pré selecção de origem polar	Não

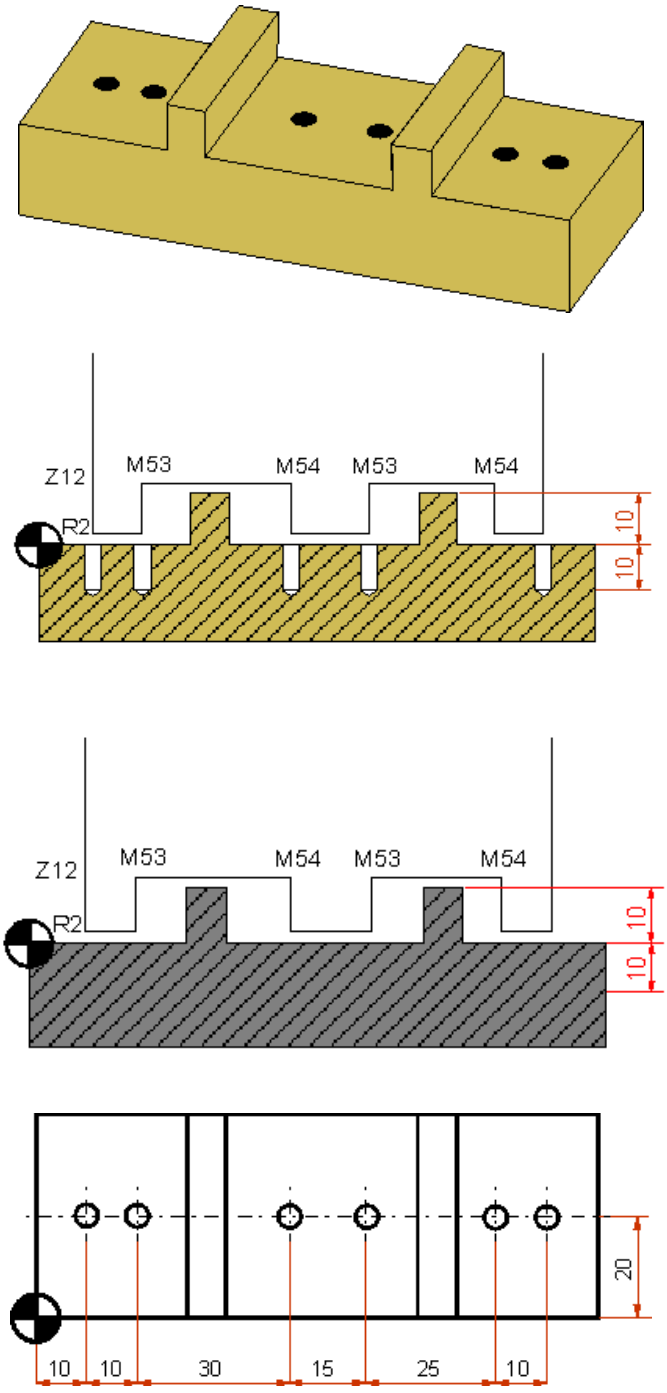
	Operadores aritméticos e relacionais	
Nome	**Significado**	
+	Adição, soma	P1=4+4 → P1=8
-	Subtracção	P1=4-4 → P1=0
*	Multiplicação	P1=4*4 → P1=16
/	Divisão	P1=4/4 → P1=1
EQ	Igual	
NE	Diferente	
GT	Maior que	
GE	Maior ou igual que	
LT	Menor que	
LE	Menor ou igual que	

Funções M	
Nome	**Significado**
M00	Paragem do programa
M01	Paragem condicional do programa
M02	Fim de programa
M03	Rotação da ferramenta à direita
M04	Rotação da ferramenta à esquerda
M05	Paragem da rotação da ferramenta
M06	Troca automática da ferramenta
M08	Activar refrigerante
M09	Desactivar refrigerante
M19	Paragem orientada da ferramenta
M30	Fim programa com retorno ao início, anula todas MS
M41	Gamas de velocidade da ferramenta
M42	Gamas de velocidade da ferramenta
M43	Gamas de velocidade da ferramenta
M44	Gamas de velocidade da ferramenta

	Funções trigonométricas	
Nome	**Significado**	
SIN	Seno	P1=SIN 30 → P1=0.5
COS	Co-seno	P1=COS 30 → P1=0.86
TAN	Tangente	P1=TAN 30 → P1=0.57
ASIN	Arco Seno	P1=ASIN 1 → P1=90
ACOS	Arco co-seno	P1=ACOS 1 → P1=0
ATAN	Arco tangente	P1=ATAN 1 → P1=45

B.6 Exemplo

Pretende-se neste exemplo mostrar como escrever o programa peça que permite maquinar a peça mostrada na figura seguinte. Esta peça é uma das peças disponíveis na base de dados do *Broker* e que um *Cliente* pode encomendar e maquinar num dos recursos existentes nos vários *Produtores*.



Ao longo do programa vão ser documentadas as linhas do programa que se acharem mais importantes para que a escrita deste programa seja entendida de uma forma clara.

N10 G90	Programação em cotas absolutas
N20 G15 H01	Eixo perpendicular ao eixo de trabalho
N30 G71 Z12	Definição sistema de medida (métrico)
N40 T29 M06	Carregamento da ferramenta n.º29
N50 G56 F40 H29 S800 M03	Definições das condições furacão
N60 NCYL G81 Z-3 R2 F40	Definição do ciclo de furacão simples
N70 X10 Y20	Posicionamento e furacão M54 (cotas X e Y)
N80 X20	Posicionamento e furacão M53 (cotas X e Y)
N90 X50	Posicionamento e furação M54 (cotas X e Y)
N100 X65	Posicionamento e furação M53 (cotas X e Y)
N110 X90	Posicionamento e furação M54 (cotas X e Y)
N120 X100	Posicionamento e furacão M53 (cotas X e Y)
N130 G80	Anulação dos ciclos de furação
N140 T28 M6	Carregamento da ferramenta n.º28
N150 G56 Z300 H28 F30 S1500 M03	Definição do ciclo de furacão simples
N160 NCYL G73 R2 Z-11.2 Q7	Posicionamento e furacão M54 (cotas X e Y)
N170 X10 Y20	Posicionamento e furacão M53 (cotas X e Y)
N180 X20	Posicionamento e furação M54 (cotas X e Y)
N190 X50	Posicionamento e furação M53 (cotas X e Y)
N200 X65	Posicionamento e furação M54 (cotas X e Y)
N210 X90	Posicionamento e furacão M53 (cotas X e Y)
N220 X100	Anulação dos ciclos de furação
N230 G80	Anulação dos ciclos de furação
N240 G00 Z400	Posicionamento rápido na cota Z
N250 Y400	Posicionamento rápido na cota Y
N260 M30	Fim de programa

Apêndice C - MENSAGENS PROPOSTAS

Nestes anexos pretende-se mostrar o tipo de mensagens (HTML e SOAP) que foram propostas, entre as várias entidades envolvidas, no capítulo 3.5 desta dissertação. Na Figura C-1 pode ser visto o tipo, o número e o nome das mensagens que são trocadas entre as várias entidades. Nos sub capítulos seguintes mostram-se mais detalhadamente os formatos e parâmetros que constituem cada uma das mensagens.

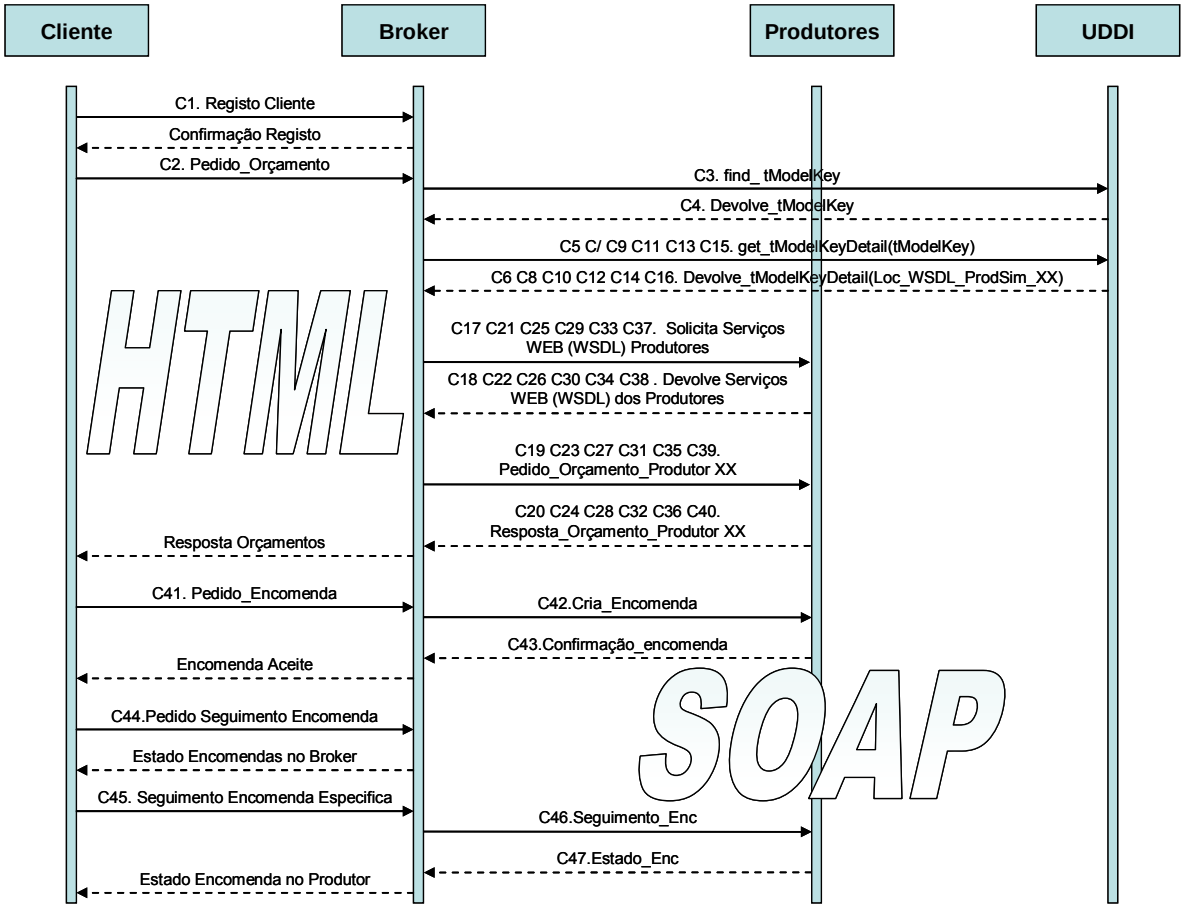


Figura C-1 – Mensagens trocadas entre as entidades nas várias fases

C.1 Mensagem HTML do Cliente para o Broker "Registo Cliente"

POST /broker/clientes.php HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, */*
Referer: http://localhost/broker/index_ficheiros/Page353.htm
Accept-Language: pt
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322)
Host: localhost
Content-Length: 137
Connection: Keep-Alive
Cache-Control: no-cache

nome=Hugo&apelido=Alvarinhas&morada=Aveiro&pais=Portugal&email=hugo.alvarinhas@netvisao.pt&bi=11058490&contrib=222052961&telef=967058528

C.2 Mensagem HTML do Cliente para o Broker "Pedido_Orçamento"

POST /broker/somaclient.php?main=editar&status=enviar HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, */*
Referer: http://localhost/broker/index_ficheiros/page396.php
Accept-Language: pt
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322)
Host: localhost
Content-Length: 32
Connection: Keep-Alive
Cache-Control: no-cache

n_peca=1&quantidade=10&pref=Custo

C.3 Mensagem SOAP do Broker para o Servidor UDDI "find_tModelKey"

```
POST /services/uddi/testregistry/inquiryapi HTTP/1.0
Date: Thu,  3 Nov 2005 20:15:22 GMT
Content-Type: text/xml; charset=UTF-8
User-agent: phpUDDI/0.03 php/5.0.1
Host: www-3.ibm.com
SOAPAction: ""
Content-Length: 269

<?xml version="1.0" encoding="utf-8"?>
 <Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
   <Body>
      <find_tModel  maxRows="50" xmlns="urn:uddi-org:api_v2" generic="2.0">
       <findQualifiers>
         <findQualifier>sortByDateDesc</findQualifier>
       </findQualifiers>
       <name >%Produtores UA%</name>
      </find_tModel>
    </Body>
  </Envelope>
```

C.4 Mensagem SOAP do Servidor UDDI para o Broker "Devolve_tModelKey"

```
HTTP/1.1 200 OK
Via: HTTP/1.1 www-3.ibm.com (IBM-PROXY-WTE)
Date: Thu, 03 Nov 2005 19:56:06 GMT
Server: IBM_HTTP_SERVER/1.3.28  Apache/1.3.28 (Unix)
Content-Length: 1657
Content-Type: text/xml
Content-Language: en-US

<?xml version="1.0" encoding="UTF-8" ?>
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <tModelList generic="2.0" xmlns="urn:uddi-org:api_v2" operator="www.ibm.com/services/uddi"
truncated="false">

      <tModelInfos>
        <tModelInfo tModelKey="UUID:96ABC580-4BBE-11DA-8A45-000629DC0A53">
          <name>Produtor Simulado 82</name>
        </tModelInfo>
        <tModelInfo tModelKey="UUID:96ABC580-4BBE-11DA-8A45-000629DC0A54">
          <name>Produtor Simulado 83</name>
        </tModelInfo>
         <tModelInfo tModelKey="UUID:96ABC580-4BBE-11DA-8A45-000629DC0A55">
          <name> Produtor Simulado 84</name>
        </tModelInfo>
        <tModelInfo tModelKey="UUID:96ABC580-4BBE-11DA-8A45-000629DC0A56">
          <name> Produtor Simulado 85</name>
        </tModelInfo>
        <tModelInfo tModelKey="UUID:96ABC580-4BBE-11DA-8A45-000629DC0A57">
          <name> Produtor Simulado 86</name>
        </tModelInfo>
        <tModelInfo tModelKey="UUID:8301E1D0-0C3C-11D9-81EB-000629DC0A53">
          <name>Produtor U.A. Heidenhein</name>
        </tModelInfo>
      </tModelInfos>
    </tModelList>
  </SOAP:Body>
</SOAP:Envelope>
```

C.5 Mensagem SOAP do Broker para o Servidor UDDI "get_tModelKeyDetail (tModeKey_ProdSim82)"

```
POST /services/uddi/testregistry/inquiryapi HTTP/1.0
Date: Thu,  3 Nov 2005 20:01:25 GMT
Content-Type: text/xml; charset=UTF-8
User-agent: phpUDDI/0.03 php/5.0.1
Host: www-3.ibm.com
SOAPAction: ""
Content-Length: 266

<?xml version="1.0" encoding="utf-8"?>
  <Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
   <Body>
     <get_tModelDetail  xmlns="urn:uddi-org:api_v2" generic="2.0">
       <tModelKey> UUID: 96ABC580-4BBE-11DA-8A45-000629DC0A53</tModelKey>
    </get_tModelDetail>
  </Body>
</Envelope>
```

C.6 Mensagem SOAP do Servidor UDDI para o Broker "Devolve_tModelKeyDetail(Loc_WSDL_ProdSim82)"

HTTP/1.1 200 OK
Via: HTTP/1.1 www-3.ibm.com (IBM-PROXY-WTE)
Date: Thu, 03 Nov 2005 20:02:04 GMT
Server: IBM_HTTP_SERVER/1.3.28 Apache/1.3.28 (Unix)
Content-Length: 744
Content-Type: text/xml
Content-Language: en-US

```
<?xml version="1.0" encoding="UTF-8" ?>
 <SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <tModelDetail generic="2.0" xmlns="urn:uddi-org:api_v2"
      operator="www.ibm.com/services/uddi" truncated="false">
      <tModel tModelKey="uuid: 96ABC580-4BBE-11DA-8A45-000629DC0A53 "
        operator="www.ibm.com/services/uddi" authorizedName="060000JUJQ">
          <name>Produtor Simulado 82</name>
          <description xml:lang="pt">Maquinar Peças</description>
         <overviewDoc>
           <overviewURL>
                    http://ims.mec.ua.pt/broker/servidor82/servidor82.wsdl
           </overviewURL>
         </overviewDoc>
         <categoryBag>
           <keyedReference tModelKey="UUID:D1ACH26F-9652-4604-9770-39B756E62AB4"
             keyName="Unique identifier" keyValue="identifier">
           </keyedReference>
         </categoryBag>
    </tModel>
  </tModelDetail>
 </SOAP:Body>
</SOAP:Envelope>
```

C.7 Mensagem SOAP do Broker para o Servidor UDDI "get_tModelKeyDetail (tModeKey_ProdSim83)"

```
POST /services/uddi/testregistry/inquiryapi HTTP/1.0
Date: Thu,  3 Nov 2005 20:01:25 GMT
Content-Type: text/xml; charset=UTF-8
User-agent: phpUDDI/0.03 php/5.0.1
Host: www-3.ibm.com
SOAPAction: ""
Content-Length: 266

<?xml version="1.0" encoding="utf-8"?>
  <Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
   <Body>
     <get_tModelDetail  xmlns="urn:uddi-org:api_v2" generic="2.0">
       <tModelKey> UUID: 96ABC580-4BBE-11DA-8A45-000629DC0A54</tModelKey>
    </get_tModelDetail>
  </Body>
</Envelope>
```

C.8 Mensagem SOAP do Servidor UDDI para o Broker "Devolve_tModelKeyDetail(Loc_WSDL_ProdSim83)"

```
HTTP/1.1 200 OK
Via: HTTP/1.1 www-3.ibm.com (IBM-PROXY-WTE)
Date: Thu, 03 Nov 2005 20:02:04 GMT
Server: IBM_HTTP_SERVER/1.3.28  Apache/1.3.28 (Unix)
Content-Length: 744
Content-Type: text/xml
Content-Language: en-US

<?xml version="1.0" encoding="UTF-8" ?>
 <SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <tModelDetail generic="2.0" xmlns="urn:uddi-org:api_v2"
      operator="www.ibm.com/services/uddi" truncated="false">
      <tModel tModelKey="uuid: 96ABC580-4BBE-11DA-8A45-000629DC0A54"
        operator="www.ibm.com/services/uddi" authorizedName="060000JUJQ">
          <name>Produtor Simulado 83</name>
          <description xml:lang="pt">Maquinar Peças</description>
         <overviewDoc>
            <overviewURL>
                  http://ims.mec.ua.pt/broker/servidor83/servidor83.wsdl
           </overviewURL>
         </overviewDoc>
         <categoryBag>
          <keyedReference tModelKey="UUID:K5ACR24D-9672-4404-9K70-39B756E62AB4"
            keyName="Unique identifier" keyValue="identifier">
          </keyedReference>
        </categoryBag>
    </tModel>
  </tModelDetail>
 </SOAP:Body>
</SOAP:Envelope>
```

C.9 Mensagem SOAP do Broker para o Servidor UDDI "get_tModelKeyDetail (tModeKey_ProdSim84)"

```
POST /services/uddi/testregistry/inquiryapi HTTP/1.0
Date: Thu,  3 Nov 2005 20:01:25 GMT
Content-Type: text/xml; charset=UTF-8
User-agent: phpUDDI/0.03 php/5.0.1
Host: www-3.ibm.com
SOAPAction: ""
Content-Length: 266

<?xml version="1.0" encoding="utf-8"?>
  <Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
   <Body>
     <get_tModelDetail  xmlns="urn:uddi-org:api_v2" generic="2.0">
       <tModelKey> UUID: 96ABC580-4BBE-11DA-8A45-000629DC0A55</tModelKey>
    </get_tModelDetail>
  </Body>
</Envelope>
```

C.10 Mensagem SOAP do Servidor UDDI para o Broker "Devolve_tModelKeyDetail(Loc_WSDL_ProdSim84)"

HTTP/1.1 200 OK
Via: HTTP/1.1 www-3.ibm.com (IBM-PROXY-WTE)
Date: Thu, 03 Nov 2005 20:02:04 GMT
Server: IBM_HTTP_SERVER/1.3.28 Apache/1.3.28 (Unix)
Content-Length: 744
Content-Type: text/xml
Content-Language: en-US

```
<?xml version="1.0" encoding="UTF-8" ?>
 <SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <tModelDetail generic="2.0" xmlns="urn:uddi-org:api_v2"
      operator="www.ibm.com/services/uddi" truncated="false">
      <tModel tModelKey="uuid: 96ABC580-4BBE-11DA-8A45-000629DC0A55"
        operator="www.ibm.com/services/uddi" authorizedName="060000JUJQ">
          <name>Produtor Simulado 84</name>
          <description xml:lang="pt">Maquinar Peças</description>
         <overviewDoc>
            <overviewURL>
                  http://ims.mec.ua.pt/broker/servidor84/servidor84.wsdl
           </overviewURL>
         </overviewDoc>
         <categoryBag>
          <keyedReference tModelKey="UUID:K5ACR24D-9672-4404-9K70-39B756E62AB4"
            keyName="Unique identifier" keyValue="identifier">
          </keyedReference>
         </categoryBag>
    </tModel>
   </tModelDetail>
  </SOAP:Body>
 </SOAP:Envelope>
```

C.11 Mensagem SOAP do Broker para o Servidor UDDI "get_tModelKeyDetail (tModeKey_ProdSim85)"

```
POST /services/uddi/testregistry/inquiryapi HTTP/1.0
Date: Thu,  3 Nov 2005 20:01:25 GMT
Content-Type: text/xml; charset=UTF-8
User-agent: phpUDDI/0.03 php/5.0.1
Host: www-3.ibm.com
SOAPAction: ""
Content-Length: 266

<?xml version="1.0" encoding="utf-8"?>
  <Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
   <Body>
     <get_tModelDetail  xmlns="urn:uddi-org:api_v2" generic="2.0">
       <tModelKey> UUID: 96ABC580-4BBE-11DA-8A45-000629DC0A56</tModelKey>
    </get_tModelDetail>
  </Body>
</Envelope>
```

C.12 Mensagem SOAP do Servidor UDDI para o Broker "Devolve_tModelKeyDetail(Loc_WSDL_ProdSim85)"

HTTP/1.1 200 OK
Via: HTTP/1.1 www-3.ibm.com (IBM-PROXY-WTE)
Date: Thu, 03 Nov 2005 20:02:04 GMT
Server: IBM_HTTP_SERVER/1.3.28 Apache/1.3.28 (Unix)
Content-Length: 744
Content-Type: text/xml
Content-Language: en-US

```
<?xml version="1.0" encoding="UTF-8" ?>
 <SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <tModelDetail generic="2.0" xmlns="urn:uddi-org:api_v2"
      operator="www.ibm.com/services/uddi" truncated="false">
      <tModel tModelKey="uuid: 96ABC580-4BBE-11DA-8A45-000629DC0A56"
        operator="www.ibm.com/services/uddi" authorizedName="060000JUJQ">
          <name>Produtor Simulado 85</name>
          <description xml:lang="pt">Maquinar Peças</description>
         <overviewDoc>
            <overviewURL>
                  http://ims.mec.ua.pt/broker/servidor85/servidor85.wsdl
           </overviewURL>
         </overviewDoc>
         <categoryBag>
          <keyedReference tModelKey="UUID:K5ACR24D-9672-4404-9K70-39B756E62AB4"
            keyName="Unique identifier" keyValue="identifier">
          </keyedReference>
        </categoryBag>
    </tModel>
  </tModelDetail>
 </SOAP:Body>
</SOAP:Envelope>
```

C.13 Mensagem SOAP do Broker para o Servidor UDDI "get_tModelKeyDetail (tModeKey_ProdSim86)"

```
POST /services/uddi/testregistry/inquiryapi HTTP/1.0
Date: Thu,  3 Nov 2005 20:01:25 GMT
Content-Type: text/xml; charset=UTF-8
User-agent: phpUDDI/0.03 php/5.0.1
Host: www-3.ibm.com
SOAPAction: ""
Content-Length: 266

<?xml version="1.0" encoding="utf-8"?>
  <Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
   <Body>
     <get_tModelDetail  xmlns="urn:uddi-org:api_v2" generic="2.0">
       <tModelKey> UUID: 96ABC580-4BBE-11DA-8A45-000629DC0A57</tModelKey>
    </get_tModelDetail>
  </Body>
</Envelope>
```

C.14 Mensagem SOAP do Servidor UDDI para o Broker "Devolve_tModelKeyDetail(Loc_WSDL_ProdSim86)"

```
HTTP/1.1 200 OK
Via: HTTP/1.1 www-3.ibm.com (IBM-PROXY-WTE)
Date: Thu, 03 Nov 2005 20:02:04 GMT
Server: IBM_HTTP_SERVER/1.3.28  Apache/1.3.28 (Unix)
Content-Length: 744
Content-Type: text/xml
Content-Language: en-US

<?xml version="1.0" encoding="UTF-8" ?>
 <SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <tModelDetail generic="2.0" xmlns="urn:uddi-org:api_v2"
      operator="www.ibm.com/services/uddi" truncated="false">
      <tModel tModelKey="uuid: 96ABC580-4BBE-11DA-8A45-000629DC0A57"
        operator="www.ibm.com/services/uddi" authorizedName="060000JUJQ">
          <name>Produtor Simulado 86</name>
          <description xml:lang="pt">Maquinar Peças</description>
         <overviewDoc>
            <overviewURL>
                  http://ims.mec.ua.pt/broker/servidor86/servidor86.wsdl
           </overviewURL>
         </overviewDoc>
         <categoryBag>
          <keyedReference tModelKey="UUID:K5ACR24D-9672-4404-9K70-39B756E62AB4"
            keyName="Unique identifier" keyValue="identifier">
          </keyedReference>
        </categoryBag>
    </tModel>
  </tModelDetail>
 </SOAP:Body>
</SOAP:Envelope>
```

C.15 Mensagem SOAP do Broker para o Servidor UDDI “get_tModelKeyDetail (tModeKey_ProdReal_UA)”

```
POST /services/uddi/testregistry/inquiryapi HTTP/1.0
Date: Thu,  3 Nov 2005 20:01:25 GMT
Content-Type: text/xml; charset=UTF-8
User-agent: phpUDDI/0.03 php/5.0.1
Host: www-3.ibm.com
SOAPAction: "“
Content-Length: 266

<?xml version="1.0" encoding="utf-8"?>
  <Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
   <Body>
     <get_tModelDetail  xmlns="urn:uddi-org:api_v2" generic="2.0">
       <tModelKey> UUID: 8301E1D0-0C3C-11D9-81EB-000629DC0A53</tModelKey>
    </get_tModelDetail>
  </Body>
</Envelope>
```

C.16 Mensagem SOAP do Servidor UDDI para o Broker "Devolve_tModelKeyDetail(Loc_WSDL_ProdReal_UA)"

```
HTTP/1.1 200 OK
Via: HTTP/1.1 www-3.ibm.com (IBM-PROXY-WTE)
Date: Thu, 03 Nov 2005 20:02:04 GMT
Server: IBM_HTTP_SERVER/1.3.28  Apache/1.3.28 (Unix)
Content-Length: 744
Content-Type: text/xml
Content-Language: en-US

<?xml version="1.0" encoding="UTF-8" ?>
 <SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <tModelDetail generic="2.0" xmlns="urn:uddi-org:api_v2"
      operator="www.ibm.com/services/uddi" truncated="false">
      <tModel tModelKey="uuid:8301E1D0-0C3C-11D9-81EB-000629DC0A53"
        operator="www.ibm.com/services/uddi" authorizedName="060000JUJQ">
          <name>Produtor U.A. Heindenhein</name>
          <description xml:lang="pt">Maquinar Peças</description>
         <overviewDoc>
            <overviewURL>
                    http://ims.mec.ua.pt/broker/maq_h/maq_h.wsdl
           </overviewURL>
         </overviewDoc>
         <categoryBag>
           <keyedReference tModelKey="UUID:C1ACF26D-9672-4404-9D70-39B756E62AB4"
             keyName="Unique identifier" keyValue="identifier">
           </keyedReference>
         </categoryBag>
    </tModel>
  </tModelDetail>
 </SOAP:Body>
</SOAP:Envelope>
```

C.17 Mensagem do Broker para o Produtor Simulado 82 "Solicita Serviços WEB (WSDL) Produtor 82"

GET /broker/servidor82/servidor82.wsdl HTTP/1.0
User-Agent: NuSOAP/0.6.3
Host: ims.mec.ua.pt
Content-Type: text/xml; charset=ISO-8859-1
Content-Length: 0

C.18 Mensagem SOAP do Produtor Simulado 82 para o Broker "Devolve Serviços WEB (WSDL) Produtor 82"

HTTP/1.1 200 OK
Date: Mon, 05 Dec 2005 17:04:01 GMT
Server: Apache/2.0.50 (Win32)
Last-Modified: Mon, 05 Dec 2005 17:03:34 GMT
ETag: "ae1d-679-e63c9409"
Accept-Ranges: bytes
Content-Length: 1646
Connection: close
Content-Type: text/plain

```
<?xml version="1.0" ?>
<definitions
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:si=http://soapinterop.org/xsd
xmlns:tns="urn:seguimentowsdl"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="urn:seguimentowsdl">

<message name="orcamentoRequest">
  <part name="produto" type="xsd:string" />
  <part name="quantidade" type="xsd:string" />
  <part name="serv" type="xsd:string" />
  <part name="material" type="xsd:string" />
</message>

<message name="orcamentoResponse">
  <part name="return" type="xsd:string" />
</message>

<message name="encomendaRequest">
  <part name="name" type="xsd:string" />
  <part name="apelido" type="xsd:string" />
  <part name="morada" type="xsd:string" />
  <part name="contrib" type="xsd:string" />
  <part name="prop" type="xsd:string" />
```

```
  <part name="orcamento" type="xsd:string" />
  <part name="peca" type="xsd:string" />
  <part name="material" type="xsd:string" />
  <part name="quantidade" type="xsd:string" />
  <part name="prazo" type="xsd:string" />
</message>

<message name="encomendaResponse">
  <part name="return" type="xsd:string" />
</message>

<message name="seguimentoRequest">
  <part name="prop" type="xsd:string" />
  <part name="orcamento" type="xsd:string" />
  <part name="encomenda" type="xsd:string" />
</message>

<message name="seguimentoResponse">
  <part name="return" type="xsd:string" />
</message>

<portType name="seguimentowsdlPortType">
 <operation name="orcamento">
   <input message="tns:orcamentoRequest" />
   <output message="tns:orcamentoResponse" />
 </operation>

 <operation name="encomenda">
   <input message="tns:encomendaRequest" />
   <output message="tns:encomendaResponse" />
 </operation>

 <operation name="seguimento">
   <input message="tns:seguimentoRequest" />
   <output message="tns:seguimentoResponse" />
 </operation>
</portType>
<binding name="seguimentowsdlBinding" type="tns:seguimentowsdlPortType">
 <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
  <operation name="orcamento">
    <soap:operation soapAction="urn:orcamentowsdl#orcamento" style="rpc" />
     <input>
       <soap:body use="encoded" namespace="urn:orcamentowsdl"
         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
     </input>
```

```
    <output>
      <soap:body use="encoded" namespace="urn:orcamentowsdl"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </output>
  </operation>

  <operation name="encomenda">
    <soap:operation soapAction="urn:encomendawsdl#encomenda" style="rpc" />
    <input>
      <soap:body use="encoded" namespace="urn:encomendawsdl"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </input>
    <output>
      <soap:body use="encoded" namespace="urn:encomendawsdl"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </output>
  </operation>

  <operation name="seguimento">
    <soap:operation soapAction="urn:seguimentowsdl#seguimento" style="rpc" />
    <input>
      <soap:body use="encoded" namespace="urn:seguimentowsdl"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </input>
    <output>
      <soap:body use="encoded" namespace="urn:seguimentowsdl"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </output>
  </operation>
</binding>

<service name="seguimentowsdl">
 <port name="seguimentowsdlPort" binding="tns:seguimentowsdlBinding">
   <soap:address location="http://localhost/broker/somaserver82.php" />
  </port>
</service>
</definitions>
```

C.19 Mensagem SOAP do Broker para o Produtor Simulado 82 "Pedido_Orçamento_Produtor 82"

```
POST /broker/somaserver82.php HTTP/1.0
User-Agent: NuSOAP/0.6.3
Host: localhost
Content-Type: text/xml; charset=ISO-8859-1
Content-Length: 688
SOAPAction: "urn:orcamentowsdl#orcamento"

<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:si="http://soapinterop.org/xsd"
xmlns:nu="urn:orcamentowsdl">

 <SOAP-ENV:Body>
  <nu:orcamento>
    <produto xsi:type="xsd:string">Biela</produto>
    <quantidade xsi:type="xsd:string">10</quantidade>
    <serv xsi:type="xsd:string">Produzir</serv>
    <material xsi:type="xsd:string">Aluminio</material>
  </nu:orcamento>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

C.20 Mensagem SOAP do Produtor Simulado 82 para o Broker "Resposta_Orçamento_Produtor 82"

```
HTTP/1.1 200 OK
Date: Fri, 16 Jun 2006 15:00:01 GMT
Server: Apache/2.0.55 (Win32)
X-Powered-By: PHP/5.1.2
Connection: Close
Content-Length: 4024
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:si="http://soapinterop.org/xsd">

<SOAP-ENV:Body>
  <orcamentoResponse>
    <return xsi:type="xsd:string">
      Para Produzir 10 Biela em Aluminio demora 3 dias e tem um custo de 50 Euros (Produtor
      Simulado82)
    </return>
  </orcamentoResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

<!--
soap_server: entering parseRequest() on 16:00 2006-06-16
soap_server: method name: orcamento
soap_server: method 'orcamento' exists
soap_server: calling parser->get_response()
soap_server: parser debug:
soap_parser: Entering soap_parser()
soap_parser: found root struct orcamento, pos 2
soap_parser: adding data for scalar value produto of value Biela
soap_parser: adding data for scalar value quantidade of value 10
soap_parser: adding data for scalar value serv of value Produzir
soap_parser: adding data for scalar value material of value Aluminio
soap_parser: inside buildVal() for orcamento(pos 2) of type struct
```

soap_parser: parsed successfully, found root struct: 2 of name orcamento
soap_server: params var dump array(4) {
  ["produto"]=>
  &string(5) "Biela"
  ["quantidade"]=>
  &int(10)
  ["serv"]=>
  &string(8) "Produzir"
  ["material"]=>
  &string(8) "Aluminio"
}
soap_server: calling 'orcamento' with params
soap_server: calling method using call_user_func_array()
soap_server: response var dumpstring(98) "Para Produzir 10 Biela em Aluminio demora 3 dias e tem um custo de 50 Euros (Produtor Simulado 82)"
soap_server: done calling method: orcamento, received Para Produzir 10 Biela em Aluminio demora 3 dias e tem um custo de 50 Euros (Produtor Simulado 82) of typestring
soap_server: got a(n) string from method
soap_server: serializing return value
soap_server: return val:string(137)
"<return xsi:type="xsd:string">
    Para Produzir 10 Biela em Aluminio demora 3 dias e tem um custo de 50 Euros (Produtor Simulado 82)
</return>"
soap_server: serializing response
soap_server: WSDL debug data:
wsdl: in serializeRPCParameters with operation orcamento, direction output and 1 param(s), and xml schema version http://www.w3.org/2001/XMLSchema
wsdl: array(10) {
  ["name"]=>
  string(9) "orcamento"
  ["binding"]=>
  string(21) "seguimentowsdlBinding"
  ["endpoint"]=>
  string(40) "http://localhost/broker/somaserver82.php"
  ["soapAction"]=>
  string(27) "urn:orcamentowsdl#orcamento"
  ["style"]=>
  string(3) "rpc"
  ["input"]=>
  array(5) {
    ["use"]=>
    string(7) "encoded"
    ["namespace"]=>
    string(17) "urn:orcamentowsdl"
    ["encodingStyle"]=>

```
    string(41) "http://schemas.xmlsoap.org/soap/encoding/"
    ["message"]=>
    string(16) "orcamentoRequest"
    ["parts"]=>
    array(4) {
      ["produto"]=>
      string(10) "xsd:string"
      ["quantidade"]=>
      string(10) "xsd:string"
      ["serv"]=>
      string(10) "xsd:string"
      ["material"]=>
      string(10) "xsd:string"
    } }
  ["output"]=>
  array(5) {
    ["use"]=>
    string(7) "encoded"
    ["namespace"]=>
    string(17) "urn:orcamentowsdl"
    ["encodingStyle"]=>
    string(41) "http://schemas.xmlsoap.org/soap/encoding/"
    ["message"]=>
    string(17) "orcamentoResponse"
    ["parts"]=>
    array(1) {
      ["return"]=>
      string(10) "xsd:string"
    } }
  ["namespace"]=>
  string(17) "urn:orcamentowsdl"
  ["transport"]=>
  string(36) "http://schemas.xmlsoap.org/soap/http"
  ["documentation"]=>
  string(0) ""
}
wsdl: use=encoded
wsdl: got 1 part(s)
wsdl: serializing part "return" of type "xsd:string"
wsdl: calling serializeType w/ unnamed param
wsdl: in serializeType: return, xsd:string, Para Produzir 10 Biela em Aluminio demora 3 dias e tem um custo de 50 Euros (Produtor Simulado 82), encoded
wsdl: got a prefixed type: string, xsd
soap_server: server sending...
-->
```

C.21 Mensagem do Broker para o Produtor Simulado 83 "Solicita Serviços WEB (WSDL) Produtor 83"

GET /broker/servidor83/servidor83.wsdl HTTP/1.0
User-Agent: NuSOAP/0.6.3
Host: ims.mec.ua.pt
Content-Type: text/xml; charset=ISO-8859-1
Content-Length: 0

C.22 Mensagem SOAP do Produtor Simulado 83 para o Broker "Devolve Serviços WEB (WSDL) Produtor 83"

HTTP/1.1 200 OK
Date: Mon, 05 Dec 2005 17:04:01 GMT
Server: Apache/2.0.50 (Win32)
Last-Modified: Mon, 05 Dec 2005 17:03:34 GMT
ETag: "ae1d-679-e63c9409"
Accept-Ranges: bytes
Content-Length: 1646
Connection: close
Content-Type: text/plain

```
<?xml version="1.0" ?>
<definitions
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:si=http://soapinterop.org/xsd
xmlns:tns="urn:seguimentowsdl"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="urn:seguimentowsdl">

<message name="orcamentoRequest">
  <part name="produto" type="xsd:string" />
  <part name="quantidade" type="xsd:string" />
  <part name="serv" type="xsd:string" />
  <part name="material" type="xsd:string" />
</message>

<message name="orcamentoResponse">
  <part name="return" type="xsd:string" />
</message>

<message name="encomendaRequest">
  <part name="name" type="xsd:string" />
  <part name="apelido" type="xsd:string" />
  <part name="morada" type="xsd:string" />
  <part name="contrib" type="xsd:string" />
  <part name="prop" type="xsd:string" />
```

```
  <part name="orcamento" type="xsd:string" />
  <part name="peca" type="xsd:string" />
  <part name="material" type="xsd:string" />
  <part name="quantidade" type="xsd:string" />
  <part name="prazo" type="xsd:string" />
</message>

<message name="encomendaResponse">
  <part name="return" type="xsd:string" />
</message>

<message name="seguimentoRequest">
  <part name="prop" type="xsd:string" />
  <part name="orcamento" type="xsd:string" />
  <part name="encomenda" type="xsd:string" />
</message>

<message name="seguimentoResponse">
  <part name="return" type="xsd:string" />
</message>

<portType name="seguimentowsdlPortType">
 <operation name="orcamento">
   <input message="tns:orcamentoRequest" />
   <output message="tns:orcamentoResponse" />
 </operation>

 <operation name="encomenda">
   <input message="tns:encomendaRequest" />
   <output message="tns:encomendaResponse" />
 </operation>

 <operation name="seguimento">
   <input message="tns:seguimentoRequest" />
   <output message="tns:seguimentoResponse" />
 </operation>
</portType>
<binding name="seguimentowsdlBinding" type="tns:seguimentowsdlPortType">
 <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
  <operation name="orcamento">
    <soap:operation soapAction="urn:orcamentowsdl#orcamento" style="rpc" />
     <input>
       <soap:body use="encoded" namespace="urn:orcamentowsdl"
         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
     </input>
```

```
    <output>
      <soap:body use="encoded" namespace="urn:orcamentowsdl"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </output>
  </operation>

  <operation name="encomenda">
    <soap:operation soapAction="urn:encomendawsdl#encomenda" style="rpc" />
    <input>
      <soap:body use="encoded" namespace="urn:encomendawsdl"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </input>
    <output>
      <soap:body use="encoded" namespace="urn:encomendawsdl"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </output>
  </operation>

  <operation name="seguimento">
    <soap:operation soapAction="urn:seguimentowsdl#seguimento" style="rpc" />
    <input>
      <soap:body use="encoded" namespace="urn:seguimentowsdl"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </input>
    <output>
      <soap:body use="encoded" namespace="urn:seguimentowsdl"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </output>
  </operation>
</binding>

<service name="seguimentowsdl">
 <port name="seguimentowsdlPort" binding="tns:seguimentowsdlBinding">
   <soap:address location="http://localhost/broker/somaserver83.php" />
  </port>
</service>
</definitions>
```

C.23 Mensagem SOAP do Broker para o Produtor Simulado 83 "Pedido_Orçamento_Produtor 83"

```
POST /broker/somaserver83.php HTTP/1.0
User-Agent: NuSOAP/0.6.3
Host: localhost
Content-Type: text/xml; charset=ISO-8859-1
Content-Length: 688
SOAPAction: "urn:orcamentowsdl#orcamento"

<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:si="http://soapinterop.org/xsd"
xmlns:nu="urn:orcamentowsdl">

<SOAP-ENV:Body>
  <nu:orcamento>
     <produto xsi:type="xsd:string">Biela</produto>
     <quantidade xsi:type="xsd:string">10</quantidade>
     <serv xsi:type="xsd:string">Produzir</serv>
     <material xsi:type="xsd:string">Aluminio</material>
  </nu:orcamento>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

C.24 Mensagem SOAP do Produtor Simulado 83 para o Broker "Resposta_Orçamento_Produtor 83"

```
HTTP/1.1 200 OK
Date: Fri, 16 Jun 2006 15:00:02 GMT
Server: Apache/2.0.55 (Win32)
X-Powered-By: PHP/5.1.2
Connection: Close
Content-Length: 4029
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:si="http://soapinterop.org/xsd">

<SOAP-ENV:Body>
  <orcamentoResponse>
    <return xsi:type="xsd:string">
      Para Produzir 10 Biela em Aluminio demora 3 dias e tem um custo de 320 Euros (Produtor
      Simulado83)
     </return>
   </orcamentoResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

<!--soap_server: entering parseRequest() on 16:00 2006-06-16
soap_server: method name: orcamento
soap_server: method 'orcamento' exists
soap_server: calling parser->get_response()
soap_server: parser debug:
soap_parser: Entering soap_parser()
soap_parser: found root struct orcamento, pos 2
soap_parser: adding data for scalar value produto of value Biela
soap_parser: adding data for scalar value quantidade of value 10
soap_parser: adding data for scalar value serv of value Produzir
soap_parser: adding data for scalar value material of value Aluminio
soap_parser: inside buildVal() for orcamento(pos 2) of type struct
soap_parser: parsed successfully, found root struct: 2 of name orcamento
```

soap_server: params var dump array(4) {
["produto"]=>
&string(5) "Biela"
["quantidade"]=>
&int(10)
["serv"]=>
&string(8) "Produzir"
["material"]=>
&string(8) "Aluminio"
}
soap_server: calling 'orcamento' with params
soap_server: calling method using call_user_func_array()
soap_server: response var dumpstring(99) "Para Produzir 10 Biela em Aluminio demora 3 dias e tem um custo de 320 Euros (Produtor Simulado 83)"
soap_server: done calling method: orcamento, received Para Produzir 10 Biela em Aluminio demora 3 dias e tem um custo de 320 Euros (Produtor Simulado 83) of typestring
soap_server: got a(n) string from method
soap_server: serializing return value
soap_server: return val:string(138)
"<return xsi:type="xsd:string">
Para Produzir 10 Biela em Aluminio demora 3 dias e tem um custo de 320 Euros (Produtor
Simulado83)
</return>"
soap_server: serializing response
soap_server: WSDL debug data:
wsdl: in serializeRPCParameters with operation orcamento, direction output and 1 param(s), and xml schema version http://www.w3.org/2001/XMLSchema
wsdl: array(10) {
["name"]=>
string(9) "orcamento"
["binding"]=>
string(21) "seguimentowsdlBinding"
["endpoint"]=>
string(40) "http://localhost/broker/somaserver83.php"
["soapAction"]=>
string(27) "urn:orcamentowsdl#orcamento"
["style"]=>
string(3) "rpc"
["input"]=>
array(5) {
["use"]=>
string(7) "encoded"
["namespace"]=>
string(17) "urn:orcamentowsdl"
["encodingStyle"]=>

```
    string(41) "http://schemas.xmlsoap.org/soap/encoding/"
    ["message"]=>
    string(16) "orcamentoRequest"
    ["parts"]=>
    array(4) {
      ["produto"]=>
      string(10) "xsd:string"
      ["quantidade"]=>
      string(10) "xsd:string"
      ["serv"]=>
      string(10) "xsd:string"
      ["material"]=>
      string(10) "xsd:string"
    } }
  ["output"]=>
  array(5) {
    ["use"]=>
    string(7) "encoded"
    ["namespace"]=>
    string(17) "urn:orcamentowsdl"
    ["encodingStyle"]=>
    string(41) "http://schemas.xmlsoap.org/soap/encoding/"
    ["message"]=>
    string(17) "orcamentoResponse"
    ["parts"]=>
    array(1) {
      ["return"]=>
      string(10) "xsd:string"
    } }
  ["namespace"]=>
  string(17) "urn:orcamentowsdl"
  ["transport"]=>
  string(36) "http://schemas.xmlsoap.org/soap/http"
  ["documentation"]=>
  string(0) ""
}
wsdl: use=encoded
wsdl: got 1 part(s)
wsdl: serializing part "return" of type "xsd:string"
wsdl: calling serializeType w/ unnamed param
wsdl: in serializeType: return, xsd:string, Para Produzir 10 Biela em Aluminio demora 3 dias e tem um custo de 320 Euros (Produtor Simulado 83), encoded
wsdl: got a prefixed type: string, xsd
soap_server: server sending...
-->
```

C.25 Mensagem do Broker para o Produtor Simulado 84 "Solicita Serviços WEB (WSDL) Produtor 84"

GET /broker/servidor84/servidor84.wsdl HTTP/1.0
User-Agent: NuSOAP/0.6.3
Host: ims.mec.ua.pt
Content-Type: text/xml; charset=ISO-8859-1
Content-Length: 0

C.26 Mensagem SOAP do Produtor Simulado 84 para o Broker "Devolve Serviços WEB (WSDL) Produtor 84"

HTTP/1.1 200 OK
Date: Mon, 05 Dec 2005 17:04:01 GMT Server: Apache/2.0.50 (Win32)
Last-Modified: Mon, 05 Dec 2005 17:03:34 GMT
ETag: "ae1d-679-e63c9409"
Accept-Ranges: bytes
Content-Length: 1646
Connection: close
Content-Type: text/plain

```
<?xml version="1.0" ?>
<definitions
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:si=http://soapinterop.org/xsd
xmlns:tns="urn:seguimentowsdl"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="urn:seguimentowsdl">

<message name="orcamentoRequest">
  <part name="produto" type="xsd:string" />
  <part name="quantidade" type="xsd:string" />
  <part name="serv" type="xsd:string" />
  <part name="material" type="xsd:string" />
</message>

<message name="orcamentoResponse">
  <part name="return" type="xsd:string" />
</message>

<message name="encomendaRequest">
  <part name="name" type="xsd:string" />
  <part name="apelido" type="xsd:string" />
  <part name="morada" type="xsd:string" />
  <part name="contrib" type="xsd:string" />
  <part name="prop" type="xsd:string" />
```

```
  <part name="orcamento" type="xsd:string" />
  <part name="peca" type="xsd:string" />
  <part name="material" type="xsd:string" />
  <part name="quantidade" type="xsd:string" />
  <part name="prazo" type="xsd:string" />
</message>

<message name="encomendaResponse">
  <part name="return" type="xsd:string" />
</message>

<message name="seguimentoRequest">
  <part name="prop" type="xsd:string" />
  <part name="orcamento" type="xsd:string" />
  <part name="encomenda" type="xsd:string" />
</message>

<message name="seguimentoResponse">
  <part name="return" type="xsd:string" />
</message>

<portType name="seguimentowsdlPortType">
 <operation name="orcamento">
   <input message="tns:orcamentoRequest" />
   <output message="tns:orcamentoResponse" />
 </operation>

 <operation name="encomenda">
   <input message="tns:encomendaRequest" />
   <output message="tns:encomendaResponse" />
 </operation>

 <operation name="seguimento">
   <input message="tns:seguimentoRequest" />
   <output message="tns:seguimentoResponse" />
 </operation>
</portType>
<binding name="seguimentowsdlBinding" type="tns:seguimentowsdlPortType">
 <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
  <operation name="orcamento">
    <soap:operation soapAction="urn:orcamentowsdl#orcamento" style="rpc" />
     <input>
       <soap:body use="encoded" namespace="urn:orcamentowsdl"
         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
     </input>
```

```
    <output>
      <soap:body use="encoded" namespace="urn:orcamentowsdl"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </output>
  </operation>

  <operation name="encomenda">
   <soap:operation soapAction="urn:encomendawsdl#encomenda" style="rpc" />
    <input>
      <soap:body use="encoded" namespace="urn:encomendawsdl"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </input>
    <output>
      <soap:body use="encoded" namespace="urn:encomendawsdl"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </output>
  </operation>

  <operation name="seguimento">
   <soap:operation soapAction="urn:seguimentowsdl#seguimento" style="rpc" />
    <input>
      <soap:body use="encoded" namespace="urn:seguimentowsdl"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </input>
    <output>
      <soap:body use="encoded" namespace="urn:seguimentowsdl"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </output>
  </operation>
</binding>

<service name="seguimentowsdl">
 <port name="seguimentowsdlPort" binding="tns:seguimentowsdlBinding">
   <soap:address location="http://localhost/broker/somaserver84.php" />
  </port>
</service>
</definitions>
```

C.27 Mensagem do Broker para o Produtor Simulado 84 "Pedido_Orçamento_Produtor 84"

```
POST /broker/somaserver84.php HTTP/1.0
User-Agent: NuSOAP/0.6.3
Host: localhost
Content-Type: text/xml; charset=ISO-8859-1
Content-Length: 688
SOAPAction: "urn:orcamentowsdl#orcamento"

<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:si="http://soapinterop.org/xsd"
xmlns:nu="urn:orcamentowsdl">

 <SOAP-ENV:Body>
  <nu:orcamento>
     <produto xsi:type="xsd:string">Biela</produto>
     <quantidade xsi:type="xsd:string">10</quantidade>
     <serv xsi:type="xsd:string">Produzir</serv>
     <material xsi:type="xsd:string">Aluminio</material>
   </nu:orcamento>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

C.28 Mensagem SOAP do Produtor Simulado 84 para o Broker "Resposta_Orçamento_Produtor 84"

```
HTTP/1.1 200 OK
Date: Fri, 16 Jun 2006 15:00:02 GMT
Server: Apache/2.0.55 (Win32)
X-Powered-By: PHP/5.1.2
Connection: Close
Content-Length: 4024
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:si="http://soapinterop.org/xsd">

 <SOAP-ENV:Body>
  <orcamentoResponse>
   <return xsi:type="xsd:string">
        Para Produzir 10 Biela em Aluminio demora 2 dias e tem um custo de 70 Euros (Produtor
        Simulado84)
   </return>
  </orcamentoResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

<!--soap_server: entering parseRequest() on 16:00 2006-06-16
soap_server: method name: orcamento
soap_server: method 'orcamento' exists
soap_server: calling parser->get_response()
soap_server: parser debug:
soap_parser: Entering soap_parser()
soap_parser: found root struct orcamento, pos 2
soap_parser: adding data for scalar value produto of value Biela
soap_parser: adding data for scalar value quantidade of value 10
soap_parser: adding data for scalar value serv of value Produzir
soap_parser: adding data for scalar value material of value Aluminio
soap_parser: inside buildVal() for orcamento(pos 2) of type struct
soap_parser: parsed successfully, found root struct: 2 of name orcamento
```

```
soap_server: params var dump array(4) {
  ["produto"]=>
  &string(5) "Biela"
  ["quantidade"]=>
  &int(10)
  ["serv"]=>
  &string(8) "Produzir"
  ["material"]=>
  &string(8) "Aluminio"
}
soap_server: calling 'orcamento' with params
soap_server: calling method using call_user_func_array()
soap_server: response var dumpstring(98) "Para Produzir 10 Biela em Aluminio demora 2 dias e tem um custo de 70 Euros (Produtor Simulado 84)"
soap_server: done calling method: orcamento, received Para Produzir 10 Biela em Aluminio demora 2 dias e tem um custo de 70 Euros (Produtor Simulado 84) of typestring
soap_server: got a(n) string from method
soap_server: serializing return value
soap_server: return val:string(137) "
<return xsi:type="xsd:string">
  Para Produzir 10 Biela em Aluminio demora 2 dias e tem um custo de 70 Euros (Produtor
  Simulado84)
</return>"
soap_server: serializing response
soap_server: WSDL debug data:
wsdl: in serializeRPCParameters with operation orcamento, direction output and 1 param(s), and xml schema version http://www.w3.org/2001/XMLSchema
wsdl: array(10) {
  ["name"]=>
  string(9) "orcamento"
  ["binding"]=>
  string(21) "seguimentowsdlBinding"
  ["endpoint"]=>
  string(40) "http://localhost/broker/somaserver84.php"
  ["soapAction"]=>
  string(27) "urn:orcamentowsdl#orcamento"
  ["style"]=>
  string(3) "rpc"
  ["input"]=>
  array(5) {
    ["use"]=>
    string(7) "encoded"
    ["namespace"]=>
    string(17) "urn:orcamentowsdl"
    ["encodingStyle"]=>
```

```
    string(41) "http://schemas.xmlsoap.org/soap/encoding/"
    ["message"]=>
    string(16) "orcamentoRequest"
    ["parts"]=>
    array(4) {
      ["produto"]=>
      string(10) "xsd:string"
      ["quantidade"]=>
      string(10) "xsd:string"
      ["serv"]=>
      string(10) "xsd:string"
      ["material"]=>
      string(10) "xsd:string"
    }
  }
  ["output"]=>
  array(5) {
    ["use"]=>
    string(7) "encoded"
    ["namespace"]=>
    string(17) "urn:orcamentowsdl"
    ["encodingStyle"]=>
    string(41) "http://schemas.xmlsoap.org/soap/encoding/"
    ["message"]=>
    string(17) "orcamentoResponse"
    ["parts"]=>
    array(1) {
      ["return"]=>
      string(10) "xsd:string"
    }
  }  ["namespace"]=>
  string(17) "urn:orcamentowsdl"
  ["transport"]=>
  string(36) "http://schemas.xmlsoap.org/soap/http"
  ["documentation"]=>
  string(0) ""
}wsdl: use=encoded
wsdl: got 1 part(s)
wsdl: serializing part "return" of type "xsd:string"
wsdl: calling serializeType w/ unnamed param
wsdl: in serializeType: return, xsd:string, Para Produzir 10 Biela em Aluminio demora 2 dias e tem um custo de 70 Euros (Produtor Simulado 84), encoded
wsdl: got a prefixed type: string, xsd
soap_server: server sending...
-->
```

C.29 Mensagem do Broker para o Produtor Simulado 85 "Solicita Serviços WEB (WSDL) Produtor 85"

GET /broker/servidor85/servidor85.wsdl HTTP/1.0
User-Agent: NuSOAP/0.6.3
Host: ims.mec.ua.pt
Content-Type: text/xml; charset=ISO-8859-1
Content-Length: 0

C.30 Mensagem SOAP do Produtor Simulado 85 para o Broker "Devolve Serviços WEB (WSDL) Produtor 85"

HTTP/1.1 200 OK
Date: Mon, 05 Dec 2005 17:04:01 GMT
Server: Apache/2.0.50 (Win32)
Last-Modified: Mon, 05 Dec 2005 17:03:34 GMT
ETag: "ae1d-679-e63c9409"
Accept-Ranges: bytes
Content-Length: 1646
Connection: close
Content-Type: text/plain

```
<?xml version="1.0" ?>
<definitions
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:si=http://soapinterop.org/xsd
xmlns:tns="urn:seguimentowsdl"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="urn:seguimentowsdl">

<message name="orcamentoRequest">
  <part name="produto" type="xsd:string" />
  <part name="quantidade" type="xsd:string" />
  <part name="serv" type="xsd:string" />
  <part name="material" type="xsd:string" />
</message>

<message name="orcamentoResponse">
  <part name="return" type="xsd:string" />
</message>

<message name="encomendaRequest">
  <part name="name" type="xsd:string" />
  <part name="apelido" type="xsd:string" />
  <part name="morada" type="xsd:string" />
  <part name="contrib" type="xsd:string" />
  <part name="prop" type="xsd:string" />
```

```
  <part name="orcamento" type="xsd:string" />
  <part name="peca" type="xsd:string" />
  <part name="material" type="xsd:string" />
  <part name="quantidade" type="xsd:string" />
  <part name="prazo" type="xsd:string" />
</message>

<message name="encomendaResponse">
  <part name="return" type="xsd:string" />
</message>

<message name="seguimentoRequest">
  <part name="prop" type="xsd:string" />
  <part name="orcamento" type="xsd:string" />
  <part name="encomenda" type="xsd:string" />
</message>

<message name="seguimentoResponse">
  <part name="return" type="xsd:string" />
</message>

<portType name="seguimentowsdlPortType">
 <operation name="orcamento">
   <input message="tns:orcamentoRequest" />
   <output message="tns:orcamentoResponse" />
 </operation>

 <operation name="encomenda">
   <input message="tns:encomendaRequest" />
   <output message="tns:encomendaResponse" />
 </operation>

 <operation name="seguimento">
   <input message="tns:seguimentoRequest" />
   <output message="tns:seguimentoResponse" />
 </operation>
</portType>

<binding name="seguimentowsdlBinding" type="tns:seguimentowsdlPortType">
 <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
  <operation name="orcamento">
    <soap:operation soapAction="urn:orcamentowsdl#orcamento" style="rpc" />
     <input>
       <soap:body use="encoded" namespace="urn:orcamentowsdl"
         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
```

```
    </input>
    <output>
      <soap:body use="encoded" namespace="urn:orcamentowsdl"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </output>
  </operation>

  <operation name="encomenda">
    <soap:operation soapAction="urn:encomendawsdl#encomenda" style="rpc" />
    <input>
      <soap:body use="encoded" namespace="urn:encomendawsdl"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </input>
    <output>
      <soap:body use="encoded" namespace="urn:encomendawsdl"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </output>
  </operation>

  <operation name="seguimento">
    <soap:operation soapAction="urn:seguimentowsdl#seguimento" style="rpc" />
    <input>
      <soap:body use="encoded" namespace="urn:seguimentowsdl"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </input>
    <output>
      <soap:body use="encoded" namespace="urn:seguimentowsdl"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </output>
  </operation>
</binding>

<service name="seguimentowsdl">
 <port name="seguimentowsdlPort" binding="tns:seguimentowsdlBinding">
   <soap:address location="http://localhost/broker/somaserver85.php" />
  </port>
</service>
</definitions>
```

C.31 Mensagem SOAP do Produtor Simulado 85 para o Broker "Pedido_Orçamento_Produtor 85"

```
POST /broker/somaserver85.php HTTP/1.0
User-Agent: NuSOAP/0.6.3
Host: localhost
Content-Type: text/xml; charset=ISO-8859-1
Content-Length: 688
SOAPAction: "urn:orcamentowsdl#orcamento"

<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:si="http://soapinterop.org/xsd"  xmlns:nu="urn:orcamentowsdl">
 <SOAP-ENV:Body>
  <nu:orcamento>
     <produto xsi:type="xsd:string">Biela</produto>
     <quantidade xsi:type="xsd:string">10</quantidade>
     <serv xsi:type="xsd:string">Produzir</serv>
     <material xsi:type="xsd:string">Aluminio</material>
   </nu:orcamento>
  </SOAP-ENV:Body>
 </SOAP-ENV:Envelope>
```

C.32 Mensagem do Produtor Simulado 85 para o Broker "Resposta_Orçamento_Produtor 85"

```
HTTP/1.1 200 OK
Date: Fri, 16 Jun 2006 15:00:02 GMT
Server: Apache/2.0.55 (Win32)
X-Powered-By: PHP/5.1.2
Connection: Close
Content-Length: 4029
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:si="http://soapinterop.org/xsd">

<SOAP-ENV:Body>
  <orcamentoResponse>
    <return xsi:type="xsd:string">
      Para Produzir 10 Biela em Aluminio demora 1 dias e tem um custo de 180 Euros (Produtor
      Simulado85)
    </return>
  </orcamentoResponse>
</SOAP-ENV:Body></SOAP-ENV:Envelope>

<!--
soap_server: entering parseRequest() on 16:00 2006-06-16
soap_server: method name: orcamento
soap_server: method 'orcamento' exists
soap_server: calling parser->get_response()
soap_server: parser debug:
soap_parser: Entering soap_parser()
soap_parser: found root struct orcamento, pos 2
soap_parser: adding data for scalar value produto of value Biela
soap_parser: adding data for scalar value quantidade of value 10
soap_parser: adding data for scalar value serv of value Produzir
soap_parser: adding data for scalar value material of value Aluminio
soap_parser: inside buildVal() for orcamento(pos 2) of type struct
soap_parser: parsed successfully, found root struct: 2 of name orcamento
```

soap_server: params var dump array(4) {
["produto"]=>
&string(5) "Biela"
["quantidade"]=>
&int(10)
["serv"]=>
&string(8) "Produzir"
["material"]=>
&string(8) "Aluminio"
}
soap_server: calling 'orcamento' with params
soap_server: calling method using call_user_func_array()
soap_server: response var dumpstring(99) "Para Produzir 10 Biela em Aluminio demora 1 dias e tem um custo de 180 Euros (Produtor Simulado 85)"
soap_server: done calling method: orcamento, received Para Produzir 10 Biela em Aluminio demora 1 dias e tem um custo de 180 Euros (Produtor Simulado 85) of typestring
soap_server: got a(n) string from method
soap_server: serializing return value
soap_server: return val:string(138) "
<return xsi:type="xsd:string">
Para Produzir 10 Biela em Aluminio demora 1 dias e tem um custo de 180 Euros (Produtor Simulado 85)
</return>"
soap_server: serializing response
soap_server: WSDL debug data:
wsdl: in serializeRPCParameters with operation orcamento, direction output and 1 param(s), and xml schema version http://www.w3.org/2001/XMLSchema
wsdl: array(10) {
["name"]=>
string(9) "orcamento"
["binding"]=>
string(21) "seguimentowsdlBinding"
["endpoint"]=>
string(40) "http://localhost/broker/somaserver85.php"
["soapAction"]=>
string(27) "urn:orcamentowsdl#orcamento"
["style"]=>
string(3) "rpc"
["input"]=>
array(5) {
["use"]=>
string(7) "encoded"
["namespace"]=>
string(17) "urn:orcamentowsdl"
["encodingStyle"]=>
string(41) "http://schemas.xmlsoap.org/soap/encoding/"

```
    ["message"]=>
    string(16) "orcamentoRequest"
    ["parts"]=>
    array(4) {
      ["produto"]=>
      string(10) "xsd:string"
      ["quantidade"]=>
      string(10) "xsd:string"
      ["serv"]=>
      string(10) "xsd:string"
      ["material"]=>
      string(10) "xsd:string"
    } }
  ["output"]=>
  array(5) {
    ["use"]=>
    string(7) "encoded"
    ["namespace"]=>
    string(17) "urn:orcamentowsdl"
    ["encodingStyle"]=>
    string(41) "http://schemas.xmlsoap.org/soap/encoding/"
    ["message"]=>
    string(17) "orcamentoResponse"
    ["parts"]=>
    array(1) {
      ["return"]=>
      string(10) "xsd:string"
    } }
  ["namespace"]=>
  string(17) "urn:orcamentowsdl"
  ["transport"]=>
  string(36) "http://schemas.xmlsoap.org/soap/http"
  ["documentation"]=>
  string(0) ""
}
wsdl: use=encoded
wsdl: got 1 part(s)
wsdl: serializing part "return" of type "xsd:string"
wsdl: calling serializeType w/ unnamed param
wsdl: in serializeType: return, xsd:string, Para Produzir 10 Biela em Aluminio demora 1 dias e tem um
custo de 180 Euros (Produtor Simulado 85), encoded
wsdl: got a prefixed type: string, xsd
soap_server: server sending...

-->
```

C.33 Mensagem do Broker para o Produtor Simulado 86 "Solicita Serviços WEB (WSDL) Produtor 86"

GET /broker/servidor86/servidor86.wsdl HTTP/1.0
User-Agent: NuSOAP/0.6.3
Host: ims.mec.ua.pt
Content-Type: text/xml; charset=ISO-8859-1
Content-Length: 0

C.34 Mensagem SOAP do Produtor Simulado 86 para o Broker "Devolve Serviços WEB (WSDL) Produtor 86"

HTTP/1.1 200 OK
Date: Mon, 05 Dec 2005 17:04:01 GMT
Server: Apache/2.0.50 (Win32)
Last-Modified: Mon, 05 Dec 2005 17:03:34 GMT
ETag: "ae1d-679-e63c9409"
Accept-Ranges: bytes
Content-Length: 1646
Connection: close
Content-Type: text/plain

```
<?xml version="1.0" ?>
<definitions
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:si=http://soapinterop.org/xsd
xmlns:tns="urn:seguimentowsdl"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="urn:seguimentowsdl">

<message name="orcamentoRequest">
  <part name="produto" type="xsd:string" />
  <part name="quantidade" type="xsd:string" />
  <part name="serv" type="xsd:string" />
  <part name="material" type="xsd:string" />
</message>

<message name="orcamentoResponse">
  <part name="return" type="xsd:string" />
</message>
<message name="encomendaRequest">
  <part name="name" type="xsd:string" />
  <part name="apelido" type="xsd:string" />
  <part name="morada" type="xsd:string" />
  <part name="contrib" type="xsd:string" />
  <part name="prop" type="xsd:string" />
  <part name="orcamento" type="xsd:string" />
```

```
  <part name="peca" type="xsd:string" />
  <part name="material" type="xsd:string" />
  <part name="quantidade" type="xsd:string" />
  <part name="prazo" type="xsd:string" />
</message>

<message name="encomendaResponse">
  <part name="return" type="xsd:string" />
</message>

<message name="seguimentoRequest">
  <part name="prop" type="xsd:string" />
  <part name="orcamento" type="xsd:string" />
  <part name="encomenda" type="xsd:string" />
</message>

<message name="seguimentoResponse">
  <part name="return" type="xsd:string" />
</message>

<portType name="seguimentowsdlPortType">
 <operation name="orcamento">
   <input message="tns:orcamentoRequest" />
   <output message="tns:orcamentoResponse" />
 </operation>

 <operation name="encomenda">
   <input message="tns:encomendaRequest" />
   <output message="tns:encomendaResponse" />
 </operation>

 <operation name="seguimento">
   <input message="tns:seguimentoRequest" />
   <output message="tns:seguimentoResponse" />
 </operation>
</portType>

<binding name="seguimentowsdlBinding" type="tns:seguimentowsdlPortType">
 <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
  <operation name="orcamento">
    <soap:operation soapAction="urn:orcamentowsdl#orcamento" style="rpc" />
     <input>
       <soap:body use="encoded" namespace="urn:orcamentowsdl"
         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
     </input>
```

```
    <output>
      <soap:body use="encoded" namespace="urn:orcamentowsdl"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </output>
  </operation>

  <operation name="encomenda">
    <soap:operation soapAction="urn:encomendawsdl#encomenda" style="rpc" />
    <input>
      <soap:body use="encoded" namespace="urn:encomendawsdl"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </input>
    <output>
      <soap:body use="encoded" namespace="urn:encomendawsdl"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </output>
  </operation>

  <operation name="seguimento">
    <soap:operation soapAction="urn:seguimentowsdl#seguimento" style="rpc" />
    <input>
      <soap:body use="encoded" namespace="urn:seguimentowsdl"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </input>
    <output>
      <soap:body use="encoded" namespace="urn:seguimentowsdl"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </output>
  </operation>
</binding>

<service name="seguimentowsdl">
 <port name="seguimentowsdlPort" binding="tns:seguimentowsdlBinding">
   <soap:address location="http://localhost/broker/somaserver86.php" />
  </port>
</service>
</definitions>
```

C.35 Mensagem SOAP do Broker para o Produtor Simulado 86 "Pedido_Orçamento_Produtor 86"

```
POST /broker/somaserver86.php HTTP/1.0
User-Agent: NuSOAP/0.6.3
Host: localhost
Content-Type: text/xml; charset=ISO-8859-1
Content-Length: 688
SOAPAction: "urn:orcamentowsdl#orcamento"

<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:si="http://soapinterop.org/xsd"
xmlns:nu="urn:orcamentowsdl">

 <SOAP-ENV:Body>
  <nu:orcamento>
    <produto xsi:type="xsd:string">Biela</produto>
    <quantidade xsi:type="xsd:string">10</quantidade>
    <serv xsi:type="xsd:string">Produzir</serv>
    <material xsi:type="xsd:string">Aluminio</material>
   </nu:orcamento>
  </SOAP-ENV:Body>
 </SOAP-ENV:Envelope>
```

C.36 Mensagem SOAP do Produtor Simulado 86 para o Broker "Pedido_Orçamento_Produtor 86"

```
HTTP/1.1 200 OK
Date: Fri, 16 Jun 2006 15:00:02 GMT
Server: Apache/2.0.55 (Win32)
X-Powered-By: PHP/5.1.2
Connection: Close
Content-Length: 4029
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:si="http://soapinterop.org/xsd">

<SOAP-ENV:Body>
  <orcamentoResponse>
    <return xsi:type="xsd:string">
      Para Produzir 10 Biela em Aluminio demora 5 dias e tem um custo de 110 Euros (Produtor
      Simulado86)
    </return>
  </orcamentoResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

<!--soap_server: entering parseRequest() on 16:00 2006-06-16
soap_server: method name: orcamento
soap_server: method 'orcamento' exists
soap_server: calling parser->get_response()
soap_server: parser debug:
soap_parser: Entering soap_parser()
soap_parser: found root struct orcamento, pos 2
soap_parser: adding data for scalar value produto of value Biela
soap_parser: adding data for scalar value quantidade of value 10
soap_parser: adding data for scalar value serv of value Produzir
soap_parser: adding data for scalar value material of value Aluminio
soap_parser: inside buildVal() for orcamento(pos 2) of type struct
soap_parser: parsed successfully, found root struct: 2 of name orcamento
```

soap_server: params var dump array(4) {
["produto"]=>
&string(5) "Biela"
["quantidade"]=>
&int(10)
["serv"]=>
&string(8) "Produzir"
["material"]=>
&string(8) "Aluminio"
}
soap_server: calling 'orcamento' with params
soap_server: calling method using call_user_func_array()
soap_server: response var dumpstring(99) "Para Produzir 10 Biela em Aluminio demora 5 dias e tem um custo de 110 Euros (Produtor Simulado 86)"
soap_server: done calling method: orcamento, received Para Produzir 10 Biela em Aluminio demora 5 dias e tem um custo de 110 Euros (Produtor Simulado 86) of typestring
soap_server: got a(n) string from method
soap_server: serializing return value
soap_server: return val:string(138) "
<return xsi:type="xsd:string">
Para Produzir 10 Biela em Aluminio demora 5 dias e tem um custo de 110 Euros (Produtor
Simulado86)
</return>"
soap_server: serializing response
soap_server: WSDL debug data:
wsdl: in serializeRPCParameters with operation orcamento, direction output and 1 param(s), and xml schema version http://www.w3.org/2001/XMLSchema
wsdl: array(10) {
["name"]=>
string(9) "orcamento"
["binding"]=>
string(21) "seguimentowsdlBinding"
["endpoint"]=>
string(40) "http://localhost/broker/somaserver86.php"
["soapAction"]=>
string(27) "urn:orcamentowsdl#orcamento"
["style"]=>
string(3) "rpc"
["input"]=>
array(5) {
["use"]=>
string(7) "encoded"
["namespace"]=>
string(17) "urn:orcamentowsdl"
["encodingStyle"]=>

```
    string(41) "http://schemas.xmlsoap.org/soap/encoding/"
    ["message"]=>
    string(16) "orcamentoRequest"
    ["parts"]=>
    array(4) {
      ["produto"]=>
      string(10) "xsd:string"
      ["quantidade"]=>
      string(10) "xsd:string"
      ["serv"]=>
      string(10) "xsd:string"
      ["material"]=>
      string(10) "xsd:string"
    } }
  ["output"]=>
  array(5) {
    ["use"]=>
    string(7) "encoded"
    ["namespace"]=>
    string(17) "urn:orcamentowsdl"
    ["encodingStyle"]=>
    string(41) "http://schemas.xmlsoap.org/soap/encoding/"
    ["message"]=>
    string(17) "orcamentoResponse"
    ["parts"]=>
    array(1) {
      ["return"]=>
      string(10) "xsd:string"
    } }
  ["namespace"]=>
  string(17) "urn:orcamentowsdl"
  ["transport"]=>
  string(36) "http://schemas.xmlsoap.org/soap/http"
  ["documentation"]=>
  string(0) ""
}
wsdl: use=encoded
wsdl: got 1 part(s)
wsdl: serializing part "return" of type "xsd:string"
wsdl: calling serializeType w/ unnamed param
wsdl: in serializeType: return, xsd:string, Para Produzir 10 Biela em Aluminio demora 5 dias e tem um custo de 110 Euros (Produtor Simulado 86), encoded
wsdl: got a prefixed type: string, xsd
soap_server: server sending...
-->
```

C.37 Mensagem do Broker para o Produtor Real U.A. Heidenhein "Solicita Serviços WEB (WSDL) Produtor UA"

GET /broker/maq_h/maq_h.wsdl HTTP/1.0
User-Agent: NuSOAP/0.6.3
Host: ims.mec.ua.pt
Content-Type: text/xml; charset=ISO-8859-1
Content-Length: 0

C.38 Mensagem do Produtor Real U.A. Heidenhein para o Broker "Devolve Serviços WEB (WSDL) Produtor UA"

HTTP/1.1 200 OK
Date: Mon, 05 Dec 2005 17:04:01 GMT
Server: Apache/2.0.50 (Win32)
Last-Modified: Mon, 05 Dec 2005 17:03:34 GMT
ETag: "ae1d-679-e63c9409"
Accept-Ranges: bytes
Content-Length: 1646
Connection: close
Content-Type: text/plain

```
<?xml version="1.0" ?>
<definitions
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:si=http://soapinterop.org/xsd
xmlns:tns="urn:seguimentowsdl"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="urn:seguimentowsdl">

<message name="orcamentoRequest">
  <part name="produto" type="xsd:string" />
  <part name="quantidade" type="xsd:string" />
  <part name="serv" type="xsd:string" />
  <part name="material" type="xsd:string" />
</message>

<message name="orcamentoResponse">
  <part name="return" type="xsd:string" />
</message>

<message name="encomendaRequest">
  <part name="name" type="xsd:string" />
  <part name="apelido" type="xsd:string" />
  <part name="morada" type="xsd:string" />
  <part name="contrib" type="xsd:string" />
  <part name="prop" type="xsd:string" />
```

```
  <part name="orcamento" type="xsd:string" />
  <part name="peca" type="xsd:string" />
  <part name="material" type="xsd:string" />
  <part name="quantidade" type="xsd:string" />
  <part name="prazo" type="xsd:string" />
</message>

<message name="encomendaResponse">
  <part name="return" type="xsd:string" />
</message>

<message name="seguimentoRequest">
  <part name="prop" type="xsd:string" />
  <part name="orcamento" type="xsd:string" />
  <part name="encomenda" type="xsd:string" />
</message>

<message name="seguimentoResponse">
  <part name="return" type="xsd:string" />
</message>

<portType name="seguimentowsdlPortType">
 <operation name="orcamento">
   <input message="tns:orcamentoRequest" />
   <output message="tns:orcamentoResponse" />
 </operation>

 <operation name="encomenda">
   <input message="tns:encomendaRequest" />
   <output message="tns:encomendaResponse" />
 </operation>

 <operation name="seguimento">
   <input message="tns:seguimentoRequest" />
   <output message="tns:seguimentoResponse" />
 </operation>
</portType>

<binding name="seguimentowsdlBinding" type="tns:seguimentowsdlPortType">
 <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
  <operation name="orcamento">
    <soap:operation soapAction="urn:orcamentowsdl#orcamento" style="rpc" />
     <input>
       <soap:body use="encoded" namespace="urn:orcamentowsdl"
         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
```

```
    </input>
    <output>
      <soap:body use="encoded" namespace="urn:orcamentowsdl"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </output>
  </operation>

  <operation name="encomenda">
    <soap:operation soapAction="urn:encomendawsdl#encomenda" style="rpc" />
    <input>
      <soap:body use="encoded" namespace="urn:encomendawsdl"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </input>
    <output>
      <soap:body use="encoded" namespace="urn:encomendawsdl"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </output>
  </operation>

  <operation name="seguimento">
    <soap:operation soapAction="urn:seguimentowsdl#seguimento" style="rpc" />
    <input>
      <soap:body use="encoded" namespace="urn:seguimentowsdl"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </input>
    <output>
      <soap:body use="encoded" namespace="urn:seguimentowsdl"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </output>
  </operation>
</binding>

<service name="seguimentowsdl">
 <port name="seguimentowsdlPort" binding="tns:seguimentowsdlBinding">
   <soap:address location="http://localhost/broker/maq_h.php" />
  </port>
</service>
</definitions>
```

C.39 Mensagem SOAP do Broker para o Produtor Real U.A. Heidenhein “Pedido_Orçamento_Produtor UA”

```
POST /broker/maq_h.php HTTP/1.0
User-Agent: NuSOAP/0.6.3
Host: localhost
Content-Type: text/xml; charset=ISO-8859-1
Content-Length: 688
SOAPAction: "urn:orcamentowsdl#orcamento"

<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:si="http://soapinterop.org/xsd"
xmlns:nu="urn:orcamentowsdl">

<SOAP-ENV:Body>
  <nu:orcamento>
    <produto xsi:type="xsd:string">Biela</produto>
    <quantidade xsi:type="xsd:string">10</quantidade>
    <serv xsi:type="xsd:string">Produzir</serv>
    <material xsi:type="xsd:string">Aluminio</material>
  </nu:orcamento>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

C.40 Mensagem do Produtor Real U.A. Heidenhein para o Broker “Resposta_Orçamento_Produtor UA”

```
HTTP/1.1 200 OK
Date: Fri, 16 Jun 2006 14:31:46 GMT
Server: Apache/2.0.55 (Win32)
X-Powered-By: PHP/5.1.2
Connection: Close
Content-Length: 4043
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:si="http://soapinterop.org/xsd">

 <SOAP-ENV:Body>
  <orcamentoResponse>
   <return xsi:type="xsd:string">
       Para Produzir 10 Biela em Aluminio demora 1 dias e tem um custo de 100 Euros (Produtor U.A.
       Heidenhain)
   </return>
  </orcamentoResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

<!--soap_server: entering parseRequest() on 15:31 2006-06-16
soap_server: method name: orcamento
soap_server: method 'orcamento' exists
soap_server: calling parser->get_response()
soap_server: parser debug:
soap_parser: Entering soap_parser()
soap_parser: found root struct orcamento, pos 2
soap_parser: adding data for scalar value produto of value Biela
soap_parser: adding data for scalar value quantidade of value 10
soap_parser: adding data for scalar value serv of value Produzir
soap_parser: adding data for scalar value material of value Aluminio
soap_parser: inside buildVal() for orcamento(pos 2) of type struct
soap_parser: parsed successfully, found root struct: 2 of name orcamento
```

soap_server: params var dump array(4) {
["produto"]=>
&string(5) "Biela"
["quantidade"]=>
&int(10)
["serv"]=>
&string(8) "Produzir"
["material"]=>
&string(8) "Aluminio"
}
soap_server: calling 'orcamento' with params
soap_server: calling method using call_user_func_array()
soap_server: response var dumpstring(103) "Para Produzir 10 Biela em Aluminio demora 1 dias e tem um custo de 100 Euros (Produtor U.A. Heidenhain)"
soap_server: done calling method: orcamento, received Para Produzir 10 Biela em Aluminio demora 1 dias e tem um custo de 100 Euros (Produtor U.A. Heidenhain) of typestring
soap_server: got a(n) string from method
soap_server: serializing return value
soap_server: return val:string(142) "
<return xsi:type="xsd:string">
Para Produzir 10 Biela em Aluminio demora 1 dias e tem um custo de 100 Euros (Produtor U.A. Heidenhain)
</return>"
soap_server: serializing response
soap_server: WSDL debug data:
wsdl: in serializeRPCParameters with operation orcamento, direction output and 1 param(s), and xml schema version http://www.w3.org/2001/XMLSchema
wsdl: array(10) {
["name"]=>
string(9) "orcamento"
["binding"]=>
string(21) "seguimentowsdlBinding"
["endpoint"]=>
string(33) "http://localhost/broker/maq_h.php"
["soapAction"]=>
string(27) "urn:orcamentowsdl#orcamento"
["style"]=>
string(3) "rpc"
["input"]=>
array(5) {
["use"]=>
string(7) "encoded"
["namespace"]=>
string(17) "urn:orcamentowsdl"
["encodingStyle"]=>

```
    string(41) "http://schemas.xmlsoap.org/soap/encoding/"
    ["message"]=>
    string(16) "orcamentoRequest"
    ["parts"]=>
    array(4) {
      ["produto"]=>
      string(10) "xsd:string"
      ["quantidade"]=>
      string(10) "xsd:string"
      ["serv"]=>
      string(10) "xsd:string"
      ["material"]=>
      string(10) "xsd:string"
    } }
  ["output"]=>
  array(5) {
    ["use"]=>
    string(7) "encoded"
    ["namespace"]=>
    string(17) "urn:orcamentowsdl"
    ["encodingStyle"]=>
    string(41) "http://schemas.xmlsoap.org/soap/encoding/"
    ["message"]=>
    string(17) "orcamentoResponse"
    ["parts"]=>
    array(1) {
      ["return"]=>
      string(10) "xsd:string"
    } }
  ["namespace"]=>
  string(17) "urn:orcamentowsdl"
  ["transport"]=>
  string(36) "http://schemas.xmlsoap.org/soap/http"
  ["documentation"]=>
  string(0) ""
}
wsdl: use=encoded
wsdl: got 1 part(s)
wsdl: serializing part "return" of type "xsd:string"
wsdl: calling serializeType w/ unnamed param
wsdl: in serializeType: return, xsd:string, Para Produzir 10 Biela em Aluminio demora 1 dias e tem um custo de 100 Euros (Produtor U.A. Heidenhain), encoded
wsdl: got a prefixed type: string, xsd
soap_server: server sending...
-->
```

C.41 Mensagem HTML do Cliente para o Broker "Pedido_Encomenda"

POST /broker/encomendas.php HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, */*
Referer: http://localhost/broker/index_ficheiros/Page442.htm
Accept-Language: pt
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322)
Host: localhost
Content-Length: 100
Connection: Keep-Alive
Cache-Control: no-cache

email=hugo.alvarinhas@netvisao.pt&proposta=5&orcamento=1&pagamento=Cheque&observacao=115225613623161

C.42 Mensagem SOAP do Broker para o Produtor "Cria_Encomenda"

```
POST /broker/maq_h.php HTTP/1.0
User-Agent: NuSOAP/0.6.3
Host: localhost
Content-Type: text/xml; charset=ISO-8859-1
Content-Length: 952
SOAPAction: "urn:encomendawsdl#encomenda"

<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:si="http://soapinterop.org/xsd"
xmlns:nu="urn:encomendawsdl">

 <SOAP-ENV:Body>
  <nu:encomenda>
    <name xsi:type="xsd:string">Hugo</name>
    <apelido xsi:type="xsd:string">Gomes</apelido>
    <morada xsi:type="xsd:string">R.Mateus Fernandes</morada>
    <contrib xsi:type="xsd:string">222052961</contrib>
    <prop xsi:type="xsd:string">10</prop>
    <orcamento xsi:type="xsd:string">6</orcamento>
    <peca xsi:type="xsd:string">Biela</peca>
    <material xsi:type="xsd:string">Aluminio</material>
    <quantidade xsi:type="xsd:string">10</quantidade>
    <prazo xsi:type="xsd:string">1</prazo>
  </nu:encomenda>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

C.43 Mensagem SOAP do Produtor para o Broker "Confirmação_encomenda"

```
HTTP/1.1 200 OK
Date: Fri, 16 Jun 2006 15:39:35 GMT
Server: Apache/2.0.55 (Win32)
X-Powered-By: PHP/5.1.2
Connection: Close
Content-Length: 5008
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"  xmlns:si="http://soapinterop.org/xsd">

 <SOAP-ENV:Body>
  <encomendaResponse>
    <return xsi:type="xsd:string">
        Encomenda para Hugo Gomes - 10 Biela em Aluminio Recebida Correctamente no Produtor U.A.
        Heidenhain em 16.06.2006 16:39:35
    </return>
  </encomendaResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
<!--
soap_server: entering parseRequest() on 16:39 2006-06-16
soap_server: method name: encomenda
soap_server: method 'encomenda' exists
soap_server: calling parser->get_response()
soap_server: parser debug:
soap_parser: Entering soap_parser()
soap_parser: found root struct encomenda, pos 2
soap_parser: adding data for scalar value name of value Hugo
soap_parser: adding data for scalar value apelido of value Gomes
soap_parser: adding data for scalar value morada of value R.Mateus Fernandes
soap_parser: adding data for scalar value contrib of value 222052961
soap_parser: adding data for scalar value prop of value 10
soap_parser: adding data for scalar value orcamento of value 6
soap_parser: adding data for scalar value peca of value Biela
```

soap_parser: adding data for scalar value material of value Aluminio
soap_parser: adding data for scalar value quantidade of value 10
soap_parser: adding data for scalar value prazo of value 1
soap_parser: inside buildVal() for encomenda(pos 2) of type struct
soap_parser: parsed successfully, found root struct: 2 of name encomenda

soap_server: params var dump array(10) {
["name"]=>
&string(4) "Hugo"
["apelido"]=>
&string(5) "Gomes"
["morada"]=>
&string(18) "R.Mateus Fernandes"
["contrib"]=>
&int(222052961)
["prop"]=>
&int(10)
["orcamento"]=>
&int(6)
["peca"]=>
&string(5) "Biela"
["material"]=>
&string(8) "Aluminio"
["quantidade"]=>
&int(10)
["prazo"]=>
&int(1)
}
soap_server: calling 'encomenda' with params
soap_server: calling method using call_user_func_array()
soap_server: response var dumpstring(122) "Encomenda para Hugo Gomes - 10 Biela em Aluminio Recebida Correctamente no Produtor U.A. Heidenhain em 16.06.2006 16:39:35"
soap_server: done calling method: encomenda, received Encomenda para Hugo Gomes - 10 Biela em Aluminio Recebida Correctamente no Produtor U.A. Heidenhain em 16.06.2006 16:39:35 of typestring
soap_server: got a(n) string from method
soap_server: serializing return value
soap_server: return val:string(161)
"<return xsi:type="xsd:string">
Encomenda para Hugo Gomes - 10 Biela em Aluminio Recebida Correctamente no Produtor U.A. Heidenhain em 16.06.2006 16:39:35
</return>"
soap_server: serializing response
soap_server: WSDL debug data:
wsdl: in serializeRPCParameters with operation encomenda, direction output and 1 param(s), and xml schema version http://www.w3.org/2001/XMLSchema

```
wsdl: array(10) {
  ["name"]=>
  string(9) "encomenda"
  ["binding"]=>
  string(21) "seguimentowsdlBinding"
  ["endpoint"]=>
  string(33) "http://localhost/broker/maq_h.php"
  ["soapAction"]=>
  string(27) "urn:encomendawsdl#encomenda"
  ["style"]=>
  string(3) "rpc"
  ["input"]=>
  array(5) {
    ["use"]=>
    string(7) "encoded"
    ["namespace"]=>
    string(17) "urn:encomendawsdl"
    ["encodingStyle"]=>
    string(41) "http://schemas.xmlsoap.org/soap/encoding/"
    ["message"]=>
    string(16) "encomendaRequest"
    ["parts"]=>
    array(10) {
      ["name"]=>
      string(10) "xsd:string"
      ["apelido"]=>
      string(10) "xsd:string"
      ["morada"]=>
      string(10) "xsd:string"
      ["contrib"]=>
      string(10) "xsd:string"
      ["prop"]=>
      string(10) "xsd:string"
      ["orcamento"]=>
      string(10) "xsd:string"
      ["peca"]=>
      string(10) "xsd:string"
      ["material"]=>
      string(10) "xsd:string"
      ["quantidade"]=>
      string(10) "xsd:string"
      ["prazo"]=>
      string(10) "xsd:string"
    } }
  ["output"]=>
```

```
  array(5) {
    ["use"]=>
    string(7) "encoded"
    ["namespace"]=>
    string(17) "urn:encomendawsdl"
    ["encodingStyle"]=>
    string(41) "http://schemas.xmlsoap.org/soap/encoding/"
    ["message"]=>
    string(17) "encomendaResponse"
    ["parts"]=>
    array(1) {
      ["return"]=>
      string(10) "xsd:string"
    } }
  ["namespace"]=>
  string(17) "urn:encomendawsdl"
  ["transport"]=>
  string(36) "http://schemas.xmlsoap.org/soap/http"
  ["documentation"]=>
  string(0) ""
}
wsdl: use=encoded
wsdl: got 1 part(s)
wsdl: serializing part "return" of type "xsd:string"
wsdl: calling serializeType w/ unnamed param
wsdl: in serializeType: return, xsd:string, Encomenda para Hugo Gomes - 10 Biela em Aluminio Recebida
Correctamente no Produtor U.A. Heidenhain em 16.06.2006 16:39:35, encoded
wsdl: got a prefixed type: string, xsd
soap_server: server sending...
-->
```

C.44 Mensagem HTML do Cliente para o Broker "Pedido Seguimento Encomenda"

POST /broker/seguimento.php HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, */*
Referer: http://localhost/broker/index_ficheiros/Page524.htm
Accept-Language: pt
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322)
Host: localhost
Content-Length: 51
Connection: Keep-Alive
Cache-Control: no-cache

email=hugo.alvarinhas@netvisao.pt&contrib=222052961

C.45 Mensagem HTML do Cliente para o Broker "Pedido Seguimento Encomenda Específica"

POST /broker/seguimento1.php HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, */*
Referer: http://localhost/broker/seguimento.php
Accept-Language: pt
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322)
Host: localhost
Content-Length: 12
Connection: Keep-Alive
Cache-Control: no-cache

encomenda=99

C.46 Mensagem SOAP do Broker para o Produtor "Seguimento_Enc"

```
POST /broker/maq_h.php HTTP/1.0
User-Agent: NuSOAP/0.6.3
Host: localhost
Content-Type: text/xml; charset=ISO-8859-1
Content-Length: 636
SOAPAction: "urn:seguimentowsdl#seguimento"

<?xml version="1.0" encoding="ISO-8859-1"?>´
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:si="http://soapinterop.org/xsd"
xmlns:tns="urn:seguimentowsdl">

<SOAP-ENV:Body>
  <tns:seguimento>
    <prop xsi:type="xsd:string">10</prop>
    <orcamento xsi:type="xsd:string">6</orcamento>
    <encomenda xsi:type="xsd:string">160</encomenda>
  </tns:seguimento>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

C.47 Mensagem SOAP do Produtor para o Broker "Estado_Enc"

```
HTTP/1.1 200 OK
Date: Fri, 16 Jun 2006 15:45:31 GMT
Server: Apache/2.0.55 (Win32)
X-Powered-By: PHP/5.1.2
Connection: Close
Content-Length: 3893
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"  xmlns:si="http://soapinterop.org/xsd">
 <SOAP-ENV:Body>
  <seguimentoResponse>
   <return xsi:type="xsd:string">
       Estado da Encomenda 160 de broker10_6 no Produtor U.A. Heidenhain em 16.06.2006 16:45:31 ->
       Fila de Espera
   </return>
  </seguimentoResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

<!--
soap_server: entering parseRequest() on 16:45 2006-06-16
soap_server: method name: seguimento
soap_server: method 'seguimento' exists
soap_server: calling parser->get_response()
soap_server: parser debug:
soap_parser: Entering soap_parser()
soap_parser: found root struct seguimento, pos 2
soap_parser: adding data for scalar value prop of value 10
soap_parser: adding data for scalar value orcamento of value 6
soap_parser: adding data for scalar value encomenda of value 160
soap_parser: inside buildVal() for seguimento(pos 2) of type struct
soap_parser: parsed successfully, found root struct: 2 of name seguimento

soap_server: params var dump array(3) {
```

```
  ["prop"]=>
  &int(10)
  ["orcamento"]=>
  &int(6)
  ["encomenda"]=>
  &int(160)
}
```

soap_server: calling 'seguimento' with params
soap_server: calling method using call_user_func_array()
soap_server: response var dumpstring(106) "Estado da Encomenda 160 de broker10_6 no Produtor U.A. Heidenhain em 16.06.2006 16:45:31 -> Fila de Espera"

soap_server: done calling method: seguimento, received Estado da Encomenda 160 de broker10_6 no Produtor U.A. Heidenhain em 16.06.2006 16:45:31 -> Fila de Espera of typestring
soap_server: got a(n) string from method
soap_server: serializing return value
soap_server: return val:string(148) "<return xsi:type="xsd:string">Estado da Encomenda 160 de broker10_6 no Produtor U.A. Heidenhain em 16.06.2006 16:45:31 -> Fila de Espera</return>"

soap_server: serializing response
soap_server: WSDL debug data:
wsdl: in serializeRPCParameters with operation seguimento, direction output and 1 param(s), and xml schema version http://www.w3.org/2001/XMLSchema

```
wsdl: array(10) {
  ["name"]=>
  string(10) "seguimento"
  ["binding"]=>
  string(21) "seguimentowsdlBinding"
  ["endpoint"]=>
  string(33) "http://localhost/broker/maq_h.php"
  ["soapAction"]=>
  string(29) "urn:seguimentowsdl#seguimento"
  ["style"]=>
  string(3) "rpc"
  ["input"]=>
  array(5) {
    ["use"]=>
    string(7) "encoded"
    ["namespace"]=>
    string(18) "urn:seguimentowsdl"
    ["encodingStyle"]=>
    string(41) "http://schemas.xmlsoap.org/soap/encoding/"
    ["message"]=>
    string(17) "seguimentoRequest"
```

```
    ["parts"]=>
    array(3) {
      ["prop"]=>
      string(10) "xsd:string"
      ["orcamento"]=>
      string(10) "xsd:string"
      ["encomenda"]=>
      string(10) "xsd:string"
    }
  }
  ["output"]=>
  array(5) {
    ["use"]=>
    string(7) "encoded"
    ["namespace"]=>
    string(18) "urn:seguimentowsdl"
    ["encodingStyle"]=>
    string(41) "http://schemas.xmlsoap.org/soap/encoding/"
    ["message"]=>
    string(18) "seguimentoResponse"
    ["parts"]=>
    array(1) {
      ["return"]=>
      string(10) "xsd:string"
    }
  }
  ["namespace"]=>
  string(18) "urn:seguimentowsdl"
  ["transport"]=>
  string(36) "http://schemas.xmlsoap.org/soap/http"
  ["documentation"]=>
  string(0) ""
}

wsdl: use=encoded
wsdl: got 1 part(s)
wsdl: serializing part "return" of type "xsd:string"
wsdl: calling serializeType w/ unnamed param
wsdl: in serializeType: return, xsd:string, Estado da Encomenda 160 de broker10_6 no Produtor U.A. Heidenhain em 16.06.2006 16:45:31 -> Fila de Espera, encoded
wsdl: got a prefixed type: string, xsd

soap_server: server sending...

-->
```